In The Name Of Allah

Book:

Lets Learn C++

By:

Ahmed Salah
Faculty Of Engineering
Electrical Eng. Department
Power & Machines Section

Email:

d1e.h4rd@yahoo.com

المحتويات:-

العناصر الاساسيه التى تتكون منها لغه c++ :-

1- جميع حروف وارقام اللغه الانجليزيه.
2- الكلمات المحجوزه مثل for int case cin void cout ..
3- الرموز الخاصه مثل @ # $ % ^ * = + - ! ..
4- المتغيرات والثوابت..
5- التعابير الرياضيه سواء حسابيه او منطقيه او مقارنه..

=========

انواع التعابير الرياضيه:-

اولا: تعابير حسابيه: وهى التى تستخدم المعاملات الحسابيه مثل الجمع او الطرح او الضرب او القسمه او باقى القسمه الصحيحه او الزياده بمقدار واحد او التناقص بمقدار واحد وذلك للربط بين اجزاءها ويكون الناتج عدد ما ..

ثانيا: تعابير منطقيه: وهى التى تستخدم المعاملات المنطقيه مثل and او or او not وذلك للربط بين اجزاءها ويكون الناتج اما صواب true(1) او خطأ false(0) ..

ثالثا: تعابير مقارنه: وهى التى تستخدم معاملات المقارنه مثل اكبر من او اصغر من او يساوى او لا يساوى او اكبر من مع يساوى او اصغر من مع يساوى ..

=========

رموز التعابير الرياضيه:-

or ||
and &&
not !
++ زياده بمقدار واحد
-- تناقص بمقدار واحد
% باقى القسمه الصحيحه
== يساوى
!= لا يساوى
> اكبر من
< اصغر من
<= اصغر من او يساوى
>= اكبر من او يساوى

=========

ملاحظات حول التعابير الرياضيه:-

اولا: معامل باقى القسمه الصحيحه وهو % يقصد به ان تجد باقى قسمه العدد الموجود على يساره على العدد الموجود على يمينه ويجب ان يكون ناتج القسمه عدد صحيح ويجب ان تكون القيم الموجوده على يمين ويسار هذا المعامل قيم صحيحه..

ثانيا: الفرق بين معامل التعيين(=) الذى يعنى تخزين القيمه التى على يمينه فى يساره ومعامل المقارنه (==)الذى يعنى هل القيمه الموجوده على يمينه تساوى القيمه الموجوده على يساره..

=========

اولويات يجب معرفتها عند تنفيذ التعابير الرياضيه:-

وهى التى تحدد العمليه التى سوف تنفذ فى البدايه ثم التى تليها وهكذا حتى النهايه وهى كالتالى:

1- الاقواس من (الداخل) الى (الخارج)
2- (الاسس)
3- (الضرب) و(القسمه) و(باقى القسمه الصحيحه)
4- (الجمع) و(الطرح)
5- (اكبر من) او (اصغر من) او (اكبر من مع يساوى) او (اصغر من مع يساوى)
6- (يساوى) و (لا يساوى)
7- معامل التعيين (=)

=========

ملاحظات على الكلمات المحجوزه:-

1- تظهر الكلمات المحجوزه بلون ازرق عند كتابتها فى البرنامج..
2- لا يجوز ان تستخدمها الا لوظيفتها المخصصه..
3- لا يجوز ان تكون اسماء لمتغيرات..

=========

انواع المتغيرات types of variables:-

1- integer variable :-

عباره عن متغيرات عدديه صحيحه سالبه او موجبه ونلاحظ ان حجمه فى الذاكره 2 بايت ومثال له 0 , 70 , 999 , -9 , -54 ..

2- long variable & short variable :-

عباره عن متغيرات عدديه صحيحه ايضا ونلاحظ ان حجمها فى الذاكره 4 بايت ..

3- character variable:-

عباره عن متغيرات حرفيه (حرف واحد) وبين اشارتى اقتباس مفرده ' ' ونلاحظ ان حجمها فى الذاكره 1 بايت ومثال له e , d , c , b , a..

4- float variable:-

عباره عن متغيرات عشريه ذات الفاصله العائمه ونلاحظ ان حجمها فى الذاكره 4 بايت ومثال له -5.0 , 4.0 , 0.0 , 0.7 , -0.1 ..

5- double variable:-

هي نفس المتغيرات العددية العشرية ولكن يمكن تمثيلها إلى خمسة
عشر خانة ونلاحظ ان حجمه فى الذاكره 8 بايت ومثال له
.. 55.98 , 13.15 , -77.987 , 13.40

6- bool variable :-

وتأخذ قيمتان فقط وهما true , false ..

=========

الصيغه العامه لتعريف متغير ما :-

Datatype variable_name = value

حيث:-

Datatype نوع المتغير سواء كان int او float او char او double ..
Variable_name اسم المتغير وذلك يخضع لنوع المتغير السابق ادخاله ..
Value القيمه المراد تخزينها بالمتغير..

امثله:-

int number1 = 9 char 'c' = a float number2 = 0.8

=========

الصيغه العامه لتعريف ثابت ما:-

const datatype constant_name = value

حيث:-

Const يجب ان تسبق جمله التعريف للدلاله على انها ثابت ..
Datatype نوع الثابت وهى نفسها انواع المتغيرات السابق
ذكرها سواء كان int او float او char او double ..
Constant_type اسم الثابت وذلك يخضع لنوع الثابت السابق ادخاله..
Value القيمه المراد تخزينها بالثابت ولن تتغير اطلاقا..

امثله:-

const double pi = 3.14

=========

الشروط التى يجب مراعاتها عند تسميه متغير ما :-

1- يجب ان لا يبدأ بفراغ او برقم ..
2- يمكن ان يبدأ بشرطه سفلى _ ..
3- يمكن استخدام الاحرف الكبيره او الصغيره معا ..
4- لا يمكن استخدام اى من الكلمات لا يمكن ان يحتوى اسم المتغير على اى من الرموز الخاصه..
5- لا يمكن ان يحتوى على اى من الكلمات المحجوزه..
6- يفضل ان يعبر اسم المتغير عن محتواه ..

امثله:-

2name لا يجوز وذلك لانه ابتدأ برقم..
a.b لا يجوز وذلك لانه احتوى على احد الرموز الخاصه..
Su m لا يجوز وذلك لانه احتوى على فراغ..
Re#u لا يجوز وذلك لانه احتوى على رمز خاص..

=========

بعض الاختصارات المهمه:-

Y++ تعادل y=y+1 تعادل y+=1 وجميعا تعنى الزياده بمقدار واحد فقط..
y-- تعادل y=y-1 تعادل y-=1 وجميعا تعنى النقصان بمقدار واحد فقط..

=========

ابسط الاوامر والجمل المستخدمه عند كتابه اى برنامج:-

✓ **جمله الاخراج او الطباعه cout :-**

شكلها العام `cout<<output;`

وظيفتها اظهار النتائج على الشاشه.

المقصود ب output فى الصيغه العامه السابقه قد تكون ثابت او نص او تعبير رياضى او متغير او جمله بين اشارتى الاقتباس المزدوجه وخلافه.

✓ **جمله الادخال cin :-**

شكلها العام `cin>>input;`

وظيفتها اعطاء المتغيرات قيم معينه اثناء تنفيذ البرنامج.

المقصود ب input فى الصيغه العامه السابقه هى اى متغير ما تريد ان تعطيه قيمه معينه اثناء تنفيذ البرنامج لاحظ انه لا يجوز استخدام متغيرات غير معرفه فى جمله الادخال فيجب تعريفها من قبل.

✓ جمله نهايه الخط endl :-

تستخدم عاده مع جمله الطباعه لجعل المؤشر يؤشر على سطر جديد بعد طباعه جمله الطباعه الموجوده بها.

=========

ملاحظه هامه جدا:-

1- عند كتابه اى برنامج باستخدام لغه c++ يجب عليك ان تبدأ بسطرين مهمين ولا تسأل عن السبب:

```
#include<iostream.h>
void main( )
```

ثم ضع البرنامج بين تلك الاقواس:

```
{
Body of programجسم البرنامج
}
```

2- اذا اردت كتابه تعليق على اى سطر من سطر البرنامج هناك طريقتين:

الاولى هى استخدام العلامه // ثم اكتب بعدها تعليقك..

والثانيه هى ان تضع تعليقك بيت تلك الاشارات:

```
/*
Your comment  تعليقك
*/
```

ولن يلتفت اليه البرنامج او يضعه فى حساباته..

=========

الجمل الشرطيه واخذ القرارات:-

اشهر الصيغ المستخدمه:-

✓ صيغه اولى:-

```
If(condition1)
Statement1;
Else(condition2)
Statement2;
Else
Statement3;
```

الشرح:-

تعنى الصيغه السابقه انه اذا تحقق الشرط الاول نفذ الجمله الاولى,,

واذا تحقق الشرط الثانى نفذ الجمله الثانيه,,

واذا لم يتحقق اى شرط مذكور نفذ الجمله الثالثه,,

وهكذا,,

✓ صيغه ثانيه:-

```
Switch( )
{
Case condition1:
Statment1;
Break;
Case condition2:
Statment2;
Break;
Default:
Statement3;
Break;
}
```

الشرح:-

تعنى switch انك امام عده حالات فاذا ادخلت ما هو مطابق للحاله الاولى يتم تنفيذ الجمله الاولى واذا ادخلت ما هو مطابق للحاله الثانيه يتم تنفيذ الجمله الثانيه وإذا ادخلت ما هو غير مطابق لاى حاله مذكوره وذلك تبعا ل default سيتم تنفيذ الجمله الثالثه وهكذا,,

✓ صيغه ثالثه:-

```
For(intial;condition;counter)
{
Statement;
}
```

الشرح:-

هذه هى الصيغه الخاصه بالكلمه المحجوزه for حيث
Intial تشير الى امر ستقوم بنفيذه قبل اول لفه فقط فى الحلقه,,
Condition تشير الى امر ستقوم بتنفيذه قبل كل لفه تقوم بها فى الحلقه,,
Statement قد تكون اكتر من امر طبعا وستنفذهم لو كان الشرط السابق ذكره صحيح,,
Counter تشير الى امر ستقوم بتنفيذ بعد كل لفه تقوم بها فى الحلقه,,

✓ صيغه رابعه:-

```
Condition?expression1:expression2;
```

الشرح:-

يقصد بالجمله السابقه انه فى حاله وضع شرط معين condition فانك ستكشف عنه اذا كان صحيحا نفذ اول امر بعد علامه الاستفهام واذا كان خطأ نفذ ثانى امر بعد علامه الاستفهام,,

=========

امثله محلوله:-

البرنامج (رقم.1):

```
#include<iostream.h>
 void main  ( )
{
cout<<"hello all the world!!"<<endl;
cout<<"this is the programmer a7mad sala7!!"<<endl;
}
```

المخرجات:

```
hello all the world!!
this is the programmer a7mad sala7!!
```

الشرح:

باستخدام الامر cout تم طباعه الجملتين المذكورتين بين علامتى الاقتباس.
وباستخدام الامر endl يتم نهايه السطر والدخول الى سطر جديد.

=============

البرنامج (رقم.2):

```
#include<iostream.h>
( ) void main
{
int var1;
int var2;
var1 = 20;
var2 =var1 + 10;
cout<<"var1 + 10 is "<<var2<<endl;
}
```

المخرجات:

```
var1 + 10 is 30
```

الشرح:

قمنا بتعريف متغيرين من النوع int وهما على الترتيب var1 و var2 حيث قمنا باعطاء المتغير الاول قيمه تساوى 20 اما عن المتغير الثانى فاعطيناه معادله وهى انه يساوى مجموع قيمه المتغير الاول و 10 .

=============

البرنامج (رقم.3):

```
#include<iostream.h>
 void main  ( )
{
char charvar1 = 'A';
char charvar2 = '\t';
```

```
char charvar3 = 'B';
cout << charvar1;
cout << charvar2;
cout << charvar3;
}
```

المخرجات:

```
A        B
```

الشرح:

قمنا بتعريف ثلاث متغيرات من النوع char اى انهم متغيرات حرفيه وتوضع بين علامتى اقتباس مفرده حيث المتغير الاول يدعى charvar1 ويساوى الحرف A اما المتغير الثالث يدعى charvar3 ويساوى الحرف B اما المتغير الثانى فيدعى charvar2 ويساوى الاختصار \t وهذا يعنى ان تترك عشر مسافات افقيه.

=============

البرنامج (رقم.4):

```
#include<iostream.h>
 void main  ( )
{
int a = 5;
int b = 8;
++a;
++b;
cout<<"a="<<a<<endl;
cout<<"b="<<b<<endl;
{
```

المخرجات:

```
a=6
b=9
```

الشرح:

قمنا بتعريف متغيرين من النوع int وهما(a الذى قيمته تساوى 5)و(b الذى قيمته تساوى 8)والسطر (++a) فيه تكون الافضليه ل 6 ,والسطر (++b) فيه تكون الافضليه ل 9 ,وذلك عند طباعه قيمتى a و b .

=============

البرنامج (رقم.5):

```
#include<iostream.h>
 void main  ( )
{
int a = 5;
int b = 8;
++a;
b++;
cout<<"a="<<a<<endl;
cout<<"b="<<b<<endl;
{
```

المخرجات:

```
a=6
b=8
```

الشرح:

قمنا بتعريف متغيرين من النوع int وهما(a الذى قيمته تساوى 5)و(b الذى قيمته تساوى 8)والسطر (a++) فيه تكون الافضليه ل 6 ,والسطر (b++) فيه تكون الافضليه ل 8 ,وذلك عند طباعه قيمتى a و b .

=============

البرنامج (رقم.6):

```
#include<iostream.h>
 void main  ( )
{
```

```
int a = 5;
int b = 8;
cout<<"a="<<++a<<endl;
cout<<"b="<<++b<<endl;
}
```

المخرجات:

```
a=6
b=9
```

الشرح:

قمنا بتعريف متغيرين من النوع int وهما(a الذى قيمته تساوى 5)و(b الذى قيمته تساوى 8)والسطر (++a) فيه تكون الافضليه ل 6 والسطر (++b) فيه تكون الافضليه ل 9 وذلك عند طباعه قيمتى a و b .

=============

البرنامج (رقم.7):

```
#include<iostream.h>
 void main  ( )
{
int a = 5;
int b = 8;
cout<<"a="<<a++<<endl
cout<<"b="<<b++<<endl
}
```

المخرجات:

```
a=5
b=8
```

الشرح:

قمنا بتعريف متغيرين من النوع int وهما(a الذى قيمته تساوى 5)و(b الذى قيمته تساوى 8)والسطر (a++) فيه تكون الافضليه ل 5 والسطر (b++) فيه تكون الافضليه ل 8 وذلك عند طباعه قيمتى a و b .

=============

البرنامج (رقم.8):

```
#include<iostream.h>
 void main  ( )
{
double n;
cout<<"please enter the number:";
cin>>n;
if(n>0)
cout<<"the number:"<<n<<" "<<"positive"<<endl;
else if(n<0)
cout<<"the number:"<<n<<" "<<"negative"<<endl;
else
cout<<"error"<<endl;
{
```

المخرجات:

فى اول حاله:

```
please enter the number:4
the number:4 pisitive
```

فى ثانى حاله:

```
please enter the number:-4
the number:-4 negative
```

فى ثالث حاله:

```
please enter the number:0
error
```

الشرح:

هذا البرنامج للتحقق من الرقم الذى تم ادخاله سواء كان موجب ام سالب واذا كان غير ذلك يطبع لنا البرنامج خطأ اى error.
حيث قمنا بتعريف متغير جديد من النوع double وهو n, ثم يطلب منك البرنامج ادخال الرقم فتدخله بنفسك,فاذا كان الرقم اكبر من الصفر يطبع لك البرنامج الرقم موجب,فاذا كان الرقم اصغر من الصفر يطبع لك البرنامج الرقم سالب,فاذا كان الرقم ليس اكبر او اصغر من الصفر يطبع لك خطأ اى error .

=============

البرنامج (رقم.9):

```
#include<iostream.h>
#include<math.h>
void main ( )
{
double number;
double answer;
cout<<"enter the number:";
cin>>number;
answer=sqrt(number);
cout<<"the square root required:"<<answer<<endl;
}
```

المخرجات:

```
enter the number:16
the square root required:4
```

الشرح:

هذا البرنامج يستخدم فى ايجاد الجزر التربيعى لعدد ما.
حيث قمنا بتعريف متغيرين جديدين من النوع double وهما number و answer, حيث يطلب منك البرنامج بادخال عدد ما ثم تدخله انت بنفسك,ومن ثم اعطينا معادله للمتغير answer وهى تعادل الجزر التربيعى للرقم المدخل,ومن ثم يطبعه.

=============

البرنامج (رقم.10):

```
#include<iostream.h>
 void main  ( )
{
int a;
cout<<"enter the number:";
cin>>a;
cout<<"the square required:"<<a*a<<endl;
cout<<"the cube required:"<<a*a*a<<endl;
}
```

المخرجات:

```
enter the number:2
the square required:4
the cube required:8
```

الشرح:

هذا البرنامج يستخدم فى ايجاد مربع ومكعب اى عدد تدخله بنفسك.
حيث قمنا بتعريف متغير جديد يدعى a من النوع int, ويطلب منك البرنامج ادخال العدد المطلوب العمل عليه,فتدخله بنفسك,ثم يطبع لك المربع والمكعب.

=============

البرنامج (رقم.11):

```
#include<iostream>
Void main ( )
{
int a;
int b;
int c;
```

```
cout<<"enter the three degrees:"<<endl;
cin>>a;
cin>>b;
cin>>c;
cout<<"the average value for the degrees given:"<<(a+b+c)/3<<endl;
}
```

المخرجات:

```
enter the three degrees:
1
2
3
the average value for the degrees given:2
```

الشرح:

يستخدم هذا البرنامج فى طباعه متوسط ثلاث علامات اى ارقام لطالب,فقد عرفنا ثلاث متغيرات من النوع int وهم يمثلوا العلامات الثلاث للطالب وهما a,b,c, حيث تدخل انت العلامات اى الارقام بنفسك,ثم يطبع لك المتوسط بجمع الثلاث علامات اى ارقام المدخله وقسمه الناتج على 3 .

=============

البرنامج (رقم.12):

```
#include<iostream.h>
Void main ( )
{
int x;
cout<<"please enter the number:";
cin>>x;
if(x>0)
cout<<"abs:"<<x;
else if(x<0);
cout<<"abs:"<<-x;
}
```

المخرجات:

فى اول حاله:

```
please enter the number:4
abs:4
```

فى ثانى حاله:

```
please enter the number:-4
abs:4
```

الشرح:

هذا البرنامج يطبع لك مقياس اى عدد.
حيث عرفنا متغير جديد من النوع int وهو x, ثم يطلب منك البرنامج ادخال العدد المطلوب العمل عليه,فاذا كان العدد المدخل اكبر من الصفر يكون المقياس هو نفسه العدد المدخل دون تغيير اشارته,واذا كان العدد المدخل اصغر من الصفر يكون المقياس هو سالب العدد المدخل اى نقوم بتغيير اشارته.

=============

البرنامج (رقم.13):

```
#include<iostream.h>
Void main ( )
{
int x;
int y;
cout<<"enter your long:";
cin>>x;
cout<<"enter your friend's long:";
cin>>y;
if(x>y)
```

```
{
cout<<"you are longer than him"<<endl;
cout<<"the difference is:"<<x-y<<endl;
}
else if (y>x)
{
cout<<"he is longer than you"<<endl;
cout<<"the difference is:"<<y-x<<endl;
}
else
cout<<"no difference"<<endl;
}
```

المخرجات:

في اول حاله:

```
Enter your long:12
Enter your friend's long:10
You are longer than him
The difference is:2
```

في ثانى حاله:

```
Enter your long:10
Enter your friend's long:12
he is longer than you
The difference is:2
```

في ثالث حاله:

```
Enter your long:10
Enter your friend's long:10
No difference
```

الشرح:

هذا البرنامج يقارن بين طولك وطول صديقك,فيخبرك اذا كان طولك اكبر منه ويطبع لك الفرق بينك وبينه,ويخبرك اذا كان طول صديقك اكبر من طولك ويطبع لك الفرق بينه وبينه,واذا كان طولك وطوله متساوين يطبع لك لا يوجد اى فرق.

=============

البرنامج (رقم.14):

```
#include<iostream.h>
void main ( )
{
float total_income;
float bs_income;
float in_stead_of_tra;
float in_stead_of_exp;
cout<<"enter bs_income:";
cin>>bs_income;
cout<<"enter in_stead_of_exp:";
cin>>in_stead_of_exp;
in_stead_of_tra =(5*bs_income)/100;
total_income = bs_income + in_stead_of_exp + in_stead_of_tra;
cout<<"total income required:"<<total_income<<endl;
}
```

المخرجات:

```
enter bs_income:200
enter in_stead_of_exp:50
total income required:260
```

الشرح:

يستخدم هذا البرنامج فى معرفه الدخل الكلى لموظف حيث فرضنا ان الدخل الكلى total_income يساوى مجموع كل من الدخل الاساسى bs_income, بدل الخبره in_stead_of_exp,بدل النقل in_stead_of_tra,حيث يقوم بانت بادخال قيم الدخل الاساسى وبدل الخبره,اما بدل النقل فيكون مساويا ل 5 فى المائه من الدخل الاساسى.

=============

البرنامج (رقم.15):

```
#include<iostream.h>
Void main ( )
{
float radius;
float area;
const double pi = 3.14;
cout<<"the calculation of area for circle:"<<endl;
cout<<"enter radius:";
cin>>radius;
area=pi*radius*radius;
cout<<"area="<<area<<endl;
}
```

المخرجات:

```
the calculation of area for circle:
enter radius:12
area=452.16
```

الشرح:

يستخدم هذا البرنامج فى حساب مساحه الدائره بمعلوميه نصف قطرها,حيث عرفنا فى البدايه متغيرين من النوع float وهما radius , area, وثابت من النوع double وهو pi وقيمته معروفه 3.14, ثم تدخل انت بنفسك قيمه نصف القطر,ويحسب لك البرنامج المساحه وهى تساوى حاصل ضرب(نصف القطر *نصف القطر*الثابت pi)ويطبعها لك.

=============

البرنامج (رقم.16):

```
#include<iostream.h>
Void main ( )
{
float length;
float base;
float area;
cout<<"the calculation of area for triangle:"<<endl;
cout<<"enter length:";
cin>>radius;
cout<<"enter base:";
cin>>base;
area=0.5*length*base;
cout<<"area="<<area<<endl;
}
```

المخرجات:

```
the calculation of area for triangle:
enter length:10
enter base:5
area=25
```

الشرح:

هذا البرنامج يحسب لنا مساحه المثلث,ففى البدايه عرفنا عدد 3 متغيرات من النوع float,الاول الطول وهو length,الثانى القاعده وهى base,الثالث المساحه وهى area,ثم يطلب منك البرنامج ادخال قيم الطول والقاعده,ثم يحسب لك المساحه ويطبعها وهى تساوى حاصل ضرب(الطول*القاعده*0.5).

=============

البرنامج (رقم.17):

```
#include<iostream.h>
```

```
void main ( )
{
int x;
cout<<"enter number a planet:";
cin>>x;
switch (x)
{
case 1:
cout<<"its a Mercury"<<endl;
break;
case 2:
cout<<"its a Venus"<<endl;
break;
case 3:
cout<<"its a earth"<<endl;
break;
case 4:
cout<<"its a Mars"<<endl;
break;
case 5:
cout<<"its a Jupiter"<<endl;
break;
case 6:
cout<<"its a Saturn"<<endl;
break;
case 7:
cout<<"its a URANUS"<<endl;
break;
case 8:
cout<<"its a NEPTUNE"<<endl;
break;
case 9:
cout<<"its a PLUTO"<<endl;
break;
Default:
cout<<"error no plant for that number"<<endl;
break;
}
}
```

المخرجات:

فى اول حالة:

```
enter number a planet:5
its a Jupiter
```

فى ثاني حالة:

```
enter number a planet:11
error no plant for that number
```

الشرح:

هذا البرنامج يطبع لك اسم الكوكب فى المجموعه الشمسيه تبعا لقربه من الشمس مثلا تدخل له الرقم 1 يطبع لك عطارد بصفته الاقرب للشمس,وتدخل له الرقم 5 يطبع لك المشترى بصفته خامس اقرب كوكب للشمس,وهكذا,,واذا ادخلت رقم غير محصور بين 1 الى 9 يطبع لك خطأ اى error .

=============

البرنامج (رقم18):

```
#include <iostream.h>
void main ( )
```

```
{
int i;
for(i=1;i<=10;i++)
cout<<"*";
}
```

المخرجات:

```
**********
```

الشرح:

هذا البرنامج يطبع لنا علامه * عشر مرات متتاليه,باستخدام الحلقه التكراريه for ,فاول جزء من جسم الحلقه ينفذ قبل اول لفه,ثانى جزء ينفذ قبل كل لفه,ثالث جزء ينفذ بعد كل لفه,لاحظ ان هذه الحلقه التكراريه سيتم تنفيذها عشر مرات متتاليه ثم نخرج منها.

=============

البرنامج (رقم.19):

```
#include <iostream.h>
void main ( )
{
int i;
for(i=1;i<=10;i++)
cout<<i<<endl;
}
```

المخرجات:

```
1
2
3
4
5
6
7
8
9
10
```

الشرح:

هذا البرنامج يطبع لك الاعداد من 1 حتى 10 ولكن كل عدد يكون فى سطر واحد على حدى وذلك لوجود الامر endl فى جمله الاخراج cout ,وطبعا يتم ذلك باستخدام حلقه تكراريه for ,وتنفذ هذه الحلقه عدد 10 مرات ثم نخرج منها,تنقسم هذه الحلقه التكراريه الى 3 اجزاء, الاول هو الشرط الابتدائى i=1 ,الثانى هو شرط الدخول للحلقه i<=10 ,الثالث هو الخطوه التى تنفذها بعد كل لفه i++ .

=============

البرنامج (رقم.20):

```
#include <iostream.h>
void main ( )
{
char c;
cout<<"enter the case you want:";
cin>>c;
switch (c)
{
case 'R':
cout<<"Red"<<endl;
break;
case 'G':
cout<<"Green"<<endl;
break;
case 'Y':
```

```
cout<<"Yellow"<<endl;
break;
case 'B':
cout<<"Blue"<<endl;
break;
Default:
cout<<"error not found"<<endl;
break;
}
}
```

المخرجات:

في اول حاله:

```
enter the case you want:G
Green
```

في ثاني حاله:

```
enter the case you want:e
error not found
```

الشرح:

هذا البرنامج يطبع لك اسم اللون تبعا للحرف المدخل من قبل المستخدم,حيث عرفنا متغير جديد من النوع char وهو c ,ثم يطلب منك ادخال الحرف,فتدخله بنفسك,فاذا ادخلت R يطبع لك red ,واذا ادخلت G يطبع لك green ,وهكذا تكمل الحالات,,اما اذا تم ادخال حرف عشوائي دون الحالات الموجوده بالبرنامج يطبع لك الجمله التاليه error not found.

==============

البرنامج (رقم.21):

```
#include <iostream.h>
void main ( )
{
long dividend;
long divisor;
char ch;
do
{
cout<<"enter dividend:";
cin>>dividend;
cout<<"enter divisor:";
cin>>divisor;
cout<<"quotient is:"<<dividend/divisor<<endl;
cout<<"remainder is:"<<dividend%divisor<<endl;
cout<<"do another (y/n):";
cin>>ch;
}
while(ch!='n');
return;
}
```

المخرجات:

```
enter dividend:12
enter divisor:3
quotient is:4
remainder is:0
do another (y/n):y
enter dividend:12
enter divisor:11
quotient is:1
remainder is:1
```

```
do another (y/n):n
```

الشرح:

هذا البرنامج يقسم لك رقمين انت تدخلهم بنفسك ويطبع لك ناتج القسمه والمتبقى من القسمه,حيث عرفنا ثلاث متغيرات اثنان منهم يمثلان المقسوم والمقسوم عليه وهما من النوع long, والثالث من النوع char, وعن امكانيه استكمال الحلقه do while فانه فى نهايه جمله do يسألك do another (y/n): فاذا قلت n يتم الخروج من البرنامج اما اذا قلت y يعود البرنامج لينفذ جمله do مره اخرى,اى فى حاله تحقق الشرط الموجود فى جمله while نعود لاستكمال الجمل الموجوده فى do, وهكذا.

=============

البرنامج (رقم.22):

```
#include <iostream.h>
Void main ( )
{
int x;
cout<<"enter the number:";
cin>>x;
if((x>=0)&&(x%2==0))
cout<<"positive and even"<<endl;
else if((x>=0)&&(x%2!=0))
cout<<"positive and odd"<<endl;
else
cout<<"error"<<endl;
}
```

المخرجات:

فى الحاله الاولى:

```
enter the number:2
positive and even
```

فى الحاله الثانيه:

```
enter the number:3
positive and odd
```

فى الحاله الثالثه:

```
enter the number:-2
error
```

الشرح:

هذا البرنامج للكشف عن الرقم المدخل سواء كان موجب وزوجى,موجب وفردى,غير ذلك,حيث عرفنا فى البدايه متغير جديد من النوع int وهو x, ثم تدخل انت الرقم المراد فحصه,فاذا كان هذا الرقم اكبر من او يساوى الصفر وناتج قسمته على 2 يساوى صفر فان البرنامج يطبع positive and even, واذا كان الرقم اكبر من او يساوى الصفر وناتج قسمته على 2 لا يساوى صفر فان البرنامج يطبع positive and odd, واذا كان غير ذلك فان البرنامج يطبع error .

=============

البرنامج (رقم.23):

```
#include <iostream.h>
void main ( )
{
int i;
for(i=0;i<10;i++)
cout<<i<<"    "<<i*i<<endl;
}
```

المخرجات:

```
0  0
1  1
2  4
3  9
4  16
5  25
```

```
6  36
7  49
8  64
9  81
```

الشرح:

هذا البرنامج يطبع لنا العدد ومربعه من الصفر حتى التسعه,حيث يتم ذلك باستخدام حلقه تكراريه for ,ويتم تنفيذها 10 مرات,فى كل مره من تنفيذ الحلقه يطبع لنا البرنامج العدد ومربعه فى سطر بسبب وجود الامر endl,حيث القيمه الابتدائيه i=0 وتنفذ قبل اول لفه,والشرط i<10 وتنفذ قبل كل لفه,والخطوه i++ وتنفذ بعد كل لفه.

=============

التراكيب فى لغه c++ :-

التركيب هو مجموعه من المتحولات البسيطه,وهذه المتحولات ممكن ان تكون من اى نوع سواء int , float ,..... ويتكون التركيب من عده عناصر او معطيات وكل عنصر او معطى يسمى عضو التركيب ويلاحظ ايضا انها تكون مختلفه فى النوع.

✓ مثال 1:-

```
struct part
{
Int modelnumber;
Int partnumber;
Float cost;
};
```

هنا قمنا اولا بالتصريح عن تركيب او نوع جديد يسمى part
وهذا التركيب له ثلاث معطيات مختلفين فى الانواع:
الاول الرقم,الثانى الكميه, الثالث السعر الافرادى..

✓ مثال 2:-

```
struct material
{
Int quantity;
Int price;
};
```

هنا قمنا اولا بالتصريح عن تركيب او نوع جديد يسمى material
وهذا التركيب له معطيان متشابهين فى النوع:
الاول يمثل الكميه,الثانى يمثل السعر..

✓ مثال 3:-

```
Struct distance
{
Int feet;
Float inches;
};
```

هنا قمنا اولا بالتصريح عن تركيب او نوع جديد يسمى distance
وهذا التركيب له معطيان مختلفان فى النوع:
الاول يمثل وحده القياس الفييت,الثانى يمثل وحده القياس الانش,
وهى وحدات قياس انجليزيه..

==========

طريقه استخدام التركيب:-

من اجل استخدام التركيب يجب اولا التصريح به وهذا يتم باستخدام الامر struct وفقا للصيغه التاليه:

```
Struct struct_name
```

حيث:

Struct_name يمثل اسم التركيب وليكن مسافه او جزء او ماده..
Struct يمثل امر التصريح نفسه..

بعد ان قمنا بالتصريح نقوم بسرد عناصر هذا التركيب بين قوسين كالتالى:

```
{
Datatype1 varlist1;
Datatype2 varlist2;
```

```
};
```

حيث:

Datatype يمثل نوع العنصر سواء كان float ,int ,double وخلافه..

Varlist يمثل العنصر او المعطى الواحد..

=========

- **وضع قيم لعناصر التركيب:-**

بعد انشاء التركيب نقوم ضمن جسم البرنامج ضمن ( ) main باعطاء كل عنصر من عناصر التركيب قيمته وذلك وفقا لامر التكليف المستخدم

حسب الصيغه التاليه:

```
Struct_name.var_name=value;
```

Struct_name تمثل اسم متحول من نوع التركيب..

Var_name تمثل اسم عضو التركيب..

Value تمثل القيمه المطلوبه..

- ✓ **فى مثال 1 السابق:-**

بعد ان قمنا بالتصريح عن التركيب وسرد عناصره الثلاث ما بين قوسين سنقوم الان باعطاء قيم لكل عنصر كالتالى:

```
Part part1;
```

قمنا الان بالتصريح عن المتحول part1 الذى هو من نوع التركيب part, والان سنقوم باعطاء القيم لعناصر التركيب:

```
Part1.modelnumber=6244;
Part1.partnumber=373;
Part1.cost=217.55;
```

- ✓ **فى مثال 3 السابق:-**

بعد ان قمنا بالتصريح عن التركيب وسرد عناصره الاثنين ما بين قوسين سنقوم الان باعطاء قيم لكل عنصر كالتالى:-

```
Distance d1;
```

قمنا الان بالتصريح عن المتحول d1 الذى هو من نوع التركيب distance, والان سنقوم باعطاء القيم لعناصر التركيب:

```
D1.feet=4;
D1.inches=6.5;
```

=========

- **تركيب خلال تركيب:-**

لمزيد من الايضاح نورد ما يوضح كيفيه استخدام التركيب الاول distance الذى يمثل المقاييس الانجليزيه مقدره بالقدم والانش كما سنستخدم ضمنه التركيب الثانى room من اجل تمثيل طول وعرض غرفه منزليه..

```
#include <iostream.h>
//////////////////////////////////////////////////////////////////////
Struct distance
{
Int feet;
Float inches;
};
///////////////////////////////////////////////////////////////////////
Struct room
{
Distance length;
Distance width;
};
```

- ✓ **بالنسبه للتركيب الاول:-** يدعى distance وله عنصران الاول الفييت وهو من النوع int, الثانى الانش وهو من النوع float ..
- ✓ **بالنسبه للتركيب الثانى:-** يدعى room وله عنصران الاول الطول, الثانى العرض وهما من النوع او التركيب الجديد الذى تم انشاؤه distance ..

بعد ما قمنا بالتصريح عن التركيب الاول والثانى نقوم الان ضمن جسم البرنامج ضمن ( )main باعطاء كل عنصر من عناصر التركيب قيمته كالتالى:

```
Room dining;
```

قمنا الان بالتصريح عن المتحول dining الذى هو من نوع التركيب room والان سنقوم باعطاء القيم لعناصر التركيب:

```
Dining.length.feet=4;
Dining.width.feet=5;
Dining.length.inches=6.5;
Dining.width.inches=4.5;
```

==========

- **المعطيات التعداديه فى لغه c++:-**

زكرنا ان التركيب هو نوع جديد يتم تعريفه من قبل المستخدم بواسطه الامر struct, ونضيف ايضا ان النوع التعدادى هو نوع جديد من المعطيات يتم تعريفه من قبل المستخدم بواسطه الامر enum ...

==========

- **متى نحتاج الى تعريف نوع تعدادى:-**

عندما يكون لدينا قائمه من القيم تخص موضوع معين فانه بالامكان تعريفها على انها نوع جديد من المعطيات التعداديه مثل:ايام الاسبوع,اشهر السنه,الالوان..الخ

==========

- **الصيغه الخاصه بتعريف معطى تعدادى:-**

```
Enum enum_type_name {list of elements};
```

حيث:

Enum تمثل امر التصريح نفسه..

Enum_type_name يمثل اسم النوع التعدادى..

List of elements تمثل قائمه العناصر التعداديه حيث يفصل ما بين العنصر والاخر اشاره فاصله..

✓ مثال1:- التوقيت الزمنى:-

```
Enum meridian {am,pm};
```

✓ مثال2:- موديلات السيارات:-

```
Enum cars {Toyota,jeep,mazda,Mercedes};
```

✓ مثال3:- الالوان:-

```
Enum colors {red,green,blue,yellow,white,black};
```

==========

- **ملاحظات على المعطيات التعداديه:-**

1- عندما نعرف نوع تعدادى فان اول عنصر له يأخذ الرقم 0 وثانى عنصر له يأخذ الرقم 1 وثالث عنصر له يأخذ الرقم 2, ولكن هذا ايضا يمكن التحكم به بشكل افضل كالتالى:

```
Enum meridian {am,pm};
```

هنا am=0 ,, وايضا pm=1 ..

```
Enum meridian {am=1,pm};
```

هنا am=1 ,, وايضا pm=2 ..

2- المتحولات من النوع التعدادى لا يمكن ادخالها او طباعتها,بل يمكن فقط تطبيق عمليات المعالجه الاخرى عليها من تكليف ومقارنه ومثال ذلك:

```
Enum meridian {am,pm};
Meridian pm=3;
Cout<<pm<<endl;
```

لعلك تظن ان خرج البرنامج هو 3 ولكن فى الحقيقه خرج البرنامج هو 1 ..

==========

- **امثله محلوله:-**

برنامج (رقم1):

```
#include <iostream.h>
///////////////////////////////////////////
enum days { sat,sun,mon,tue,wed,thu,fri };
///////////////////////////////////////////
void main ( )
{
```

```
days day1;
days day2;
day1=mon;
day2=thu;
int diff=day2-day1;
cout<<"days between="<<diff<<endl;
if(day1<day2)
cout<<"day1 comes before day2";
return;
}
```

المخرجات:

```
days between=5
day1 comes before day2
```

الشرح:

هذا البرنامج مجرد تطبيق على المعطيات التعداديه,حيث نلاحظ فى هذا البرنامج اننا عرفنا نوع تعدادى اسمه days ,وعناصره محدده ومعروفه سلفا وهم سبعه ايام,اذن لدينا عدد 7 عناصر,حيث تم التصريح بنوع تعدادى يتم وفق الصيغه التاليه:

enum enum_type_name { list of elements };

ثم قمنا داخل جسم البرنامج بتعريف متغيرين ينتميان لهذا النوع التعدادى وهم day1,day2, وقمنا بالتعريف لمتغير جديد من النوع int وهو diff, ولاحظ ان كل عنصر من العناصر السبع للمعطيات التعداديه يأخذ رقم من الصفر حتى السته مثل sat يمثل الصفر,حتى fri يمثل السته.

============

برنامج (رقم.2):

```
include <iostream.h
/////////////////////////////////////////
enum colors { white,black,yellow,green,red };
/////////////////////////////////////////
void main ( )
{
colors color1;
colors color2;
color1=black;
color2=yellow;
cout<<"the number of the best color:"<<color1<<endl;
cout<<"the number of the bad color:"<<color2<<endl;
return;
}
```

المخرجات:

```
the number of the best color:1
the number of the bad color:2
```

الشرح:

هذا البرنامج مجرد تطبيق على المعطيات التعداديه,حيث نلاحظ فى هذا البرنامج اننا عرفنا نوع تعدادى اسمه colors ,وعناصره محدده ومعروفه سلفا وهم خمسه ايام,اذن لدينا عدد 5 عناصر,حيث تم التصريح بنوع تعدادى يتم وفق الصيغه التاليه:

enum enum_type_name { list of elements };

ثم قمنا داخل جسم البرنامج بتعريف متغيرين ينتميان لهذا النوع التعدادى وهم color1,color2, ولاحظ ان كل عنصر من العناصر الخمس للمعطيات التعداديه يأخذ رقم من الصفر حتى الاربعه مثل white يأخذ الصفر حتى red يأخذ الاربعه.

============

برنامج (رقم.3):

```
#include <iostream.h>
/////////////////////////////////////////
struct part
```

```
{
int modelnumber;
int partnumber;
float cost;
};
//////////////////////////////////////////
void main ( )
{
part part1;
part1.modelnumber =6244;
part1.partnumber =373;
part1.cost =217.55;
cout<<"model:"<<part1.modelnumber<<endl;
cout<<"part:"<<part1.partnumber<<endl;
cout<<"cost in $:"<<part1.cost<<endl;
return;
}
```

المخرجات:

```
model:6244
part:373
cost in $:217.55
```

الشرح:

هذا البرنامج عن تركيب يمثل سلعه تجاريه,فهذا البرنامج يحتوى على تركيب جديد لنوع تم تصميمه من قبل المستخدم وهو part ,وهذا التركيب الجديد مكون من عدد 3 معطيات,الاول يمثل الرقم,الثانى يمثل الكميه,الثالثه تمثل السعر,المطعى الاول من النوع int وهو modelnumber, الثانى من النوع int وهو partnumber, الثالث من النوع float وهو cost, وعند الدخول الى جسم البرنامج فاننا بدأنا بسرد متغير جديد من النوع part وهو part1, ثم قمنا باعطاء كل عنصر من الثلاث عناصر المكونه للتركيب الجديد قيمته وذلك وفقا للصيغه التاليه:

struct_name.var_name

ومثال ذلك:

part1.modelnumber

ثم اعطيناه قيمه معينه ولتكن 6244 .

============

برنامج (رقم4.):

```
#include <iostream.h>
/////////////////////////////////////////////
struct part
{
int modelnumber;
int partnumber;
float cost;
};
/////////////////////////////////////////////
void main ( )
{
part part1;
part part2;
part1.modelnumber =6244;
part1.partnumber =373;
part1.cost =217.55;
cout<<"data for first part:"<<endl;
cout<<"model1:"<<part1.modelnumber<<endl;
cout<<"part1:"<<part1.partnumber<<endl;
cout<<"cost1 in $:"<<part1.cost<<endl;
part2.modelnumber =6245;
```

```
part2.partnumber =374;
part2.cost =218.55;
cout<<"data for second part:"<<endl;
cout<<"model2:"<<part2.modelnumber<<endl;
cout<<"part2:"<<part2.partnumber<<endl;
cout<<"cost2 in $:"<<part2.cost<<endl;
return;
}
```

المخرجات:

```
data for first part:
model1:6244
part1:373
cost1 in $:217.55
data for second part:
model2:6245
part2:374
cost2 in $:218.55
```

الشرح:

هذا البرنامج عن تركيب يمثل سلعه تجاريه ولكن بصوره اكثر تعقيدا,فهذا البرنامج يحتوى على تركيب جديد لنوع تم تصميمه من قبل المستخدم وهو part ,وهذا التركيب الجديد مكون من عدد 3 معطيات,الاول يمثل الرقم,الثانى يمثل الكميه,الثالثه تمثل السعر,المطعى الاول من النوع int وهو modelnumber ,الثانى من النوع int وهو partnumber ,الثالث من النوع float وهو cost ,وعند الدخول الى جسم البرنامج فاننا بدأنا بسرد متغير جديد من النوع part وهو part1 ,واخر من نفس النوع part وهو part2 ,ثم قمنا باعطاء كل عنصر من الثلاث عناصر المكونه للتركيب الجديد قيمته وذلك وفقا للصيغه التاليه:

struct_name.var_name

ومثال ذلك:

part1.modelnumber

ثم اعطيناه قيمه معينه ولتكن 6244 .

============

برنامج (رقم5):

```
#include <iostream.h>
/////////////////////////////////////////////
struct distance
{
int feet;
float inches;
};
/////////////////////////////////////////////
void main ( )
{
distance d1;
distance d2;
distance d3;
d2.feet=11;
d2.inches=6.25;
d3.feet=0;
cout<<"enter feet:";
cin>>d1.feet;
cout<<"enter inches:";
cin>>d1.inches;
d3.inches=d1.inches+d2.inches;
if(d3.inches>=12.0)
{
d3.inches-=12.0;
```

```
d3.feet++;
 }
d3.feet+=d1.feet+d2.feet;
cout<<d1.feet<<"'-"<<d1.inches<<" ' ' +";
cout<<d2.feet<<"'-"<<d2.inches<<" ' ' +";
cout<<d3.feet<<"'-"<<d3.inches<<" ' ' + ";
return;
}
```

المخرجات:

```
enter feet:10
enter inches:6.75
10'-6.75" + 11'-6.25" =22'-1"
```

الشرح:

هذا البرنامج عن تركيب لمقياس المسافات الانكليزيه,فهو يحتوى على تركيب لنوع جديد يدعى distance, هذا النوع له عدد 2 عنصر معطيات وهما(الفييت وهو من النوع int),(والانش وهو من النوع float),ثم دخلنا الى جسم البرنامج وفيه تعريف لثلاث متغيرات جدد هم d1,d2,d3 وجميعهم من التركيب الجديد distance, وكالعاده تم اعطاء كل عنصر من العناصر قيمه محدده من خلال الصيغه:

struct_name.var_name=value

ومثال لذلك:

d2.feet=11

d3.feet=0

d2.inches=6.25

============

برنامج (رقم6):

```
#include <iostream.h>
///////////////////////////////////////////////////////////////
struct distance
{
int feet;
float inches;
};
///////////////////////////////////////////////////////////////
struct room
{
distance length;
distance width;
};
///////////////////////////////////////////////////////////////
void main ( )
{
room dining;
dining.length.feet=13;
dining.length.inches=6.5;
dining.width.feet=10;
dining.width.inches=0.0;
float l=dining.length.feet + dining.length.inches/12;
float w=dining.width.feet + dining.width.inches/12;
cout<<"dining room area is "<<l*w<<" square feet"<<endl;
return;
}
```

المخرجات:

```
dining room area is 135.417 square feet
```

الشرح:

هذا البرنامج عن تركيب بداخله تركيب,اولا تركيب جديد يدعى distance له عدد 2 عنصر معطيات الاول (الفيت من النوع int) والثانى (الانش من النوع float),ثانيا تركيب اخر جديد يدعى room له عدد 2 عنصر معطيات ايضا (الاول length من النوع الجديد distance) و(الثانى width من النوع الجديد distance) ,ثم قمنا ضمن جسم البرنامج بتعريف متغير جديد يدعى dining وهو من التركيب او النوع الجديدroom , اما عن العوامل فتم اعطاؤها قيم وفق الصيغه التاليه:

struct_name.var_name=value

ومثال ذلك:

dining.length.feet=13
dining.length.inches=6.5
dining.width.feet=10
dining.width.inches=0.0

==========

التوابع او الدوال فى لغه c++ :-

الداله بكل بساطه هى مقطعه برمجى يؤدى عمل معين ولكن هذا المقطع يكون موقعه ليس ضمن جسم الداله الرئيسيه اى ليس ضمن main ( ) وانما خارجها.

==========

دواعى استخدام الدوال او التوابع:-

1- لتسهيل كتابه البرامج الكبيره.
2- لتسهيل تتبع الاخطاء.
3- لتسهيل التعديل والتطوير من البرنامج دون الحاجه الى اعاده كتابته.
4- لجعل حجم البرنامج اصغر.

==========

كيفيه التعامل مع التابع او الداله:-

ان التعامل مع لغه c++ يتطلب ان نتعرض لثلاث افكار وهم:-

✓ التصريح بالتابع: وذلك يتم فى بدايه البرنامج وهو يبدو لنا فى الصيغه التاليه:

```
Ruturn_type_value func_name (arguments);
```

حيث:

Return_type_value تمثل نوع مخرجات البرنامج اى نوع القيمه المرجعيه وقد تكون int او float او void وخلافه.

Func_name تمثل اسم التابع وهو يتم بناء على اختيار المستخدم.

Arguments تمثل معاملات التابع اى القيم التى ستمرر له من برنامج الاستدعاء ولكن احيانا قد يكون لدينا تابع ليس له معاملات.

✓ التعريف بالتابع: وذلك يتم بعد نهايه البرنامج الرئيسى,ونلاحظ ان التعريف يحتوى الاوامر التى سيتم تنفيذها لدى استدعاء التابع,تلك الاوامر التى نتحدث عنها موجوده فى جسم الداله,وهو يبدو لنا فى الصيغه التاليه:

```
Ruturn_type_value func_name (arguments);
{
Body of the functions
}
```

✓ استدعاء التابع: وهذا يتم ضمن البرنامج الرئيسى,اى ضمن main( ),ونلاحظ ان استدعاء التابع يمكن ان يكون لمره واحده او لعده مرات وذلك بالطبع حسب الحاجه اليه,وهو يبدو لنا فى الصيغه التاليه:

```
Func_name (actual values);
```

حيث:

Func_name تمثل اسم التابع وهو يتم بناء على اختيار المستخدم.

Actual values تمثل القيم الفعليه التى سيتم تمريرها الى معاملات التابع.

==========

دمج التعريف والتصريح بالتابع:- يمكن دمج (التصريح بالتابع الذى يتم فى بدايه البرنامج) مع (التعريف بالتابع الذى يتم بعد نهايه البرنامج الرئيسى) وذلك يتم فى نفس منطقه التصريح وذلك بغرض تسهيل بنيه البرنامج.

==========

تمرير المعاملات الى التوابع:- يعرف المعامل على انه عباره عن قيمه من نوع معين كالنوع int او float وغيرها,وهذه القيمه يتم تمريرها او اعطاؤها من البرنامج الى التابع,وهذه المعاملات تتيح للتابع ان يأخذ مختلف القيم من مختلف الانواع وذلك بناء على احتياجات برنامج الاستدعاء,وقد يتم تمرير:

1- ثوابت.
2- متحولات.
3- تراكيب.

===========

- خاصيه التحميل الزائد للتوابع:- نشير هنا الى امكانيه عمل اكتر من تابع يحملوا نفس الاسم ولكن يختلفان فى نوع المعاملات والقيم الفعليه التى سيتم تمريرها الى المعاملات.

===========

- المتحولات واصناف التخزين:- ان صنف التخزين لمتحول ما يحدد اى جزء من البرنامج يستطيع الوصول اليه وكم سيطول وجوده وفيما يلى سوف نتعامل مع ثلاث اصناف تخزين وهم:

1- الصنف الساكن.static
2- الصنف الخارجى.external
3- الصنف الالى.automatic

===========

اولا:المتحولات الخارجيه: global
ان المتغيرات الخارجيه او العامه او الدوليه هى متغيرات يتم الاعلان عنها خارج اى تابع اخر وجميع توابع البرنامج بامكانها استخدامها والتعامل معها.

ثانيا:المتحولات الاليه: automatic
ان هذه الانواع من المتحولات سوف تتواجد فقط طالما ان التابع المعرف ضمنه يتم تنفيذه,وتكون مرئيه ضمن هذا التابع.

- ✓ عمر حياه المتحول الالى: يلاحظ ان المتحول الالى يتشكل اليا عند استدعاءه,ويتدمر ايضا اليا عند الخروج منه..
- ✓ ظهور المتحول الالى: لا يمكنك اعطاء قيمه لمتحول ما من دون اولا التعريف به او التصريح به ضمن جسم التابع..

ثالثا:المتحولات الساكنه: static
تتواجد هذه الانواع من المتحولات طوال مده تنفيذ البرنامج ولكنها تكون فقط مرئيه ضمن تابعها الخاص بها,وتنقسم الى نوعين:

1- المتحولات الاليه الساكنه."وهى تمتلك ظهور المتحول المحلى ودوره حياه المتحول الخارجى,ان انها مرئيه ضمن التابع المعرفه ضمنه ولكنها تبقى موجوده خلال فتره عمل البرنامج"
2- المتحولات الخارجيه الساكنه.

============

- امثله محلوله:-

برنامج (رقم.1):

```
#include <iostream.h>
/////////////////////////////////////////////////////////////////////
void printmessage ( );
/////////////////////////////////////////////////////////////////////
void main ( )
{
printmessage ( )
return;
}
/////////////////////////////////////////////////////////////////////
void printmessage ( )
{
cout<<"I am a function!!"<<endl;
cout<<"I am prepared by eng: a.salah"<<endl;
}
```

المخرجات:

```
I am a function!!
I am prepared by eng: a.salah
```

الشرح:

هذا البرنامج عن تابع بسيط,يقوم بطباعه رسالتين عند استدعاءه,وفى هذا البرنامج تم استدعاء هذا التابع مره واحده,وعلى الترتيب تم التصريح بالتابع وفق الامر التالى:

void printmessage ( );

ثم جسم التابع المكون من استدعاء للتابع ثم امر الارجاع:

```
printmessage ( )
return;
```

ثم اخيرا تم التعريف بالتابع:

```
cout<<"I am a function!!"<<endl;
cout<<"I am prepared by eng: a.salah"<<endl;
```

==========

برنامج (رقم.2):

```
#include <iostream.h>
//////////////////////////////////////////////////////////////////////
void printmessage ( )
{
cout<<"I am a function!!"<<endl;
cout<<"I am prepared by eng: a.salah"<<endl;
}

//////////////////////////////////////////////////////////////////////
void main ( )
{
printmessage ( )
return;
}
```

المخرجات:

```
I am a function!!
I am prepared by eng: a.salah
```

الشرح:

هذا البرنامج يوضح خاصيه الدمج بين التعريف والتصريح بالتابع, حيث ذكر التعريف والتصريح معا فى اول البرنامج كالتالى:

```
void printmessage ( )
{
cout<<"I am a function!!"<<endl;
cout<<"I am prepared by eng: a.salah"<<endl;
}
```

ثم يليه جسم البرنامج وفيه تم استدعاء التابع لمره واحده ايضا مع امر الارجاع كالتالى:

```
Printmessage ( )
Return;
```

==========

برنامج (رقم.3):

```
#include <iostream.h>
//////////////////////////////////////////////////////////////////////
void slashline ( )
{
for(int j=0;j<20;j++)
cout<<"*";
cout<<endl;
}
//////////////////////////////////////////////////////////////////////
void main ( )
{
slashline ( );
cout<<"student"<<"   "<<"    degree"<<endl;
slashline ( );
cout<<"ahmed salah"<<"    "<<"50"<<endl;
cout<<"ahmed samir"<<"    "<<"60"<<endl;
```

```
cout<<"mohmed azmy"<<"   "<<"70"<<endl;
cout<<"tareq bakr"<<"   "<<" 80"<<endl;
slashline ( );
return;
}
```

المخرجات:

```
*******************
student      degree
*******************
ahmed salah      50
ahmed samir      60
mohmed azmy     70
tareq bakr       80
*******************
```

الشرح:

هذا البرنامج يبين امكانيه استدعاء التابع لاكثر من مره,حيث تم استدعاء التابع عدد 3 مرات بتكرار الصيغه التاليه ضمن جسم البرنامج:

slashline ( );

ونلاحظ ايضا فى هذا البرنامج انه تم دمج التعريف والتصريف بالتابع ليظهرها معا فى بدايه البرنامج.
اما عن التابع المذكور فى البرنامج فهو يدعى slashline, ووظيفته هى اظهار الرمز * بعدد 20 مره على التوالى فى سطر واحد وذلك باستخدام الحلقه التكراريه for,وينفذ هذه الوظيفه كلما تم استدعاءه.

==========

برنامج (رقم.4):

```
#include <iostream.h>
/////////////////////////////////////////////////////////////////////
void subtraction(int,int);
/////////////////////////////////////////////////////////////////////
void main ( )
{
cout<<"the 1st result:";
subtraction(7,2);
cout<<"the 2nd result:";
subtraction(6,2);
cout<<"the 3rd result:";
subtraction(5,2);
return;
}
/////////////////////////////////////////////////////////////////////
void subtraction(int a,int b)
{
int r;
r=a-b;
cout<<r;
cout<<endl;
}
```

المخرجات:

```
the 1st result:5
the 2nd result:4
the 3rd result:3
```

الشرح:

هذا البرنامج عن استدعاء التابع اكثر من مره بقيم فعليه مختلفه,حيث اولا فى الاعلى تم التصريح بتابع جديد يدعى subtraction وهذا التابع له معاملان اى قيمتيان سيطبق عليهما التابع والتى ستمرر له من برنامج الاستدعاء وهما من النوع int كالتالى:

```
void subtraction(int,int);
```

وفى نهايه البرنامج تم التعريف بالتابع مع تسميه كلا من معامليه وهما a و b ,مع ذكر الاوامر التى سينفذها التابع عند استدعاءه كالتالى:

```
void subtraction(int a,int b)
```

ثم قمنا ضمن جسم البرنامج باستدعاء التابع لمده ثلاث مرات ولكن فى كل مره تختلف القيم الفعليه التى سيتم تمريرها الى معاملات التابع الاثنين كالتالى:

```
subtraction(7,2);
subtraction(6,2);
subtraction(5,2);
```

==========

برنامج (رقم.5):

```
#include <iostream.h>
/////////////////////////////////////////////////////////////////////
void slashline(int,char);
/////////////////////////////////////////////////////////////////////
void main ( )
{
slashline(20,'*');
cout<<"student"<<"   "<<"    degree"<<endl;
slashline(20,'#');
cout<<"ahmed salah"<<"   "<<"50"<<endl;
cout<<"ahmed samir"<<"   "<<"60"<<endl;
cout<<"mohmed azmy"<<"   "<<"70"<<endl;
cout<<"tareq bakr"<<"   "<<" 80"<<endl;
slashline(20,'$');
return;
}
/////////////////////////////////////////////////////////////////////
void slashline(int n,char ch)
{
for(int j=0;j<n;j++)
cout<<ch;
cout<<endl;
}
```

المخرجات:

```
********************
student      degree
####################
ahmed salah      50
ahmed samir      60
mohmed azmy      70
tareq bakr      80
$$$$$$$$$$$$$$$$$$$$
```

الشرح:

هذا البرنامج عن استدعاء التابع اكثر من مره بقيم فعليه مختلفه,حيث اولا فى الاعلى تم التصريح بتابع جديد يدعى slashline وهذا التابع له معاملان اى قيمتيان سيطبق عليهما التابع والتى ستمرر له من برنامج الاستدعاء وهما من النوع int و char كالتالى:

```
void slashline(int,char);
```

وفى نهايه البرنامج تم التعريف بالتابع مع تسميه كلا من معامليه وهما a و b ,مع ذكر الاوامر التى سينفذها التابع عند استدعاءه كالتالى:

```
void slashline(int n,char ch)
```

ثم قمنا ضمن جسم البرنامج باستدعاء التابع لمده ثلاث مرات ولكن فى كل مره تختلف القيم الفعليه التى سيتم تمريرها الى معاملات التابع الاثنين كالتالى:

```
slashline(20,'*');
slashline(20,'#');
slashline(20,'$');
```

==========

برنامج (رقم.6):

```
#include <iostream.h>
////////////////////////////
void operate(int,int);
void operate(float,float);
////////////////////////////
void main ( )
{
int x=5;
int y=2;
float n=30.0;
float m=2.0;
operate(x,y);
operate(n,m);
}
////////////////////////////
void operate(int a,int b)
{
int r;
r=a*b;
cout<<r<<endl;
}
////////////////////////////
void operate(float a,float b)
{
int r;
r=a/b;
cout<<r<<endl;
}
```

المخرجات:

```
10
15
```

الشرح:

هذا المثال على خاصيه التحميل الزائد للتوابع حيث يحتوى البرنامج على عدد 2 داله او تابع يحملان نفس الاسم الا وهو operate, ولكن يختلفان فى نوع المعاملات والقيم الفعليه التى سيتم تمريرها الى المعاملات,لاحظ ان معاملات التابع الاول من النوع int, بينما معاملات التابع الثانى من النوع float .

==========

برنامج (رقم.7):

```
#include <iostream.h>
/////////////////////////////////////////////
void repchar( );
void repchar(char);
void repchar(char,int);
/////////////////////////////////////////////
void main ( )
{
repchar( );
repchar('=');
```

```
repchar('$',20);
}
/////////////////////////////////////////////
void repchar ( )
{
for(int j=0;j<20;j++)
cout<<"=";
cout<<endl;
}
/////////////////////////////////////////////
void repchar(char ch)
{
for(int j=0;j<20;j++)
cout<<ch;
cout<<endl;
}
/////////////////////////////////////////////
void repchar(char ch,int n)
{
for(int j=0;j<n;j++)
cout<<ch;
cout<<endl;
}
```

المخرجات:

```
********************
====================
$$$$$$$$$$$$$$$$$$$$
```

الشرح:

هذا البرنامج عن تمرير الثوابت كقيم فعليه,حيث بدايه تم التصريح بثلاث توابع كل منهم على حدى ومن الملاحظ ان كل منهم يحمل نفس الاسم كالتالى:

```
void repchar( );
void repchar(char);
void repchar(char,int);
```

ولكن من الملاحظ ان التابع الاول بدون اى معاملات,والتابع الثانى يحتوى على معامل واحد والنوع char,والتابع الثالث يحتوى على معاملان اثنان والنوعين هما char و int.

وظيفه التابع الاول هى طباعه الرمز * بمقدار 20 مره كلما تم استدعاءه.

وظيفه التابع الثانى هى طباعه الرمز = بمقدار 20 مره كلما تم استدعاءه.

وظيفه التابع الثالث هى طباعه الرمز $ بمقدار 20 مره كلما تم استدعاءه.

==========

برنامج (رقم.8):

```
#include <iostream.h>
//////////////////////////////////////////////////////////////
void repchar(char,int);
//////////////////////////////////////////////////////////////
void main ( )
{
char chin;
int nin;
cout<<"enter a character:";
cin>>chin;
cout<<"enter number of times to repeat it:";
cin>>nin;
repchar(chin,nin);
return;
```

```
}
////////////////////////////////////////////////////////////////
void repchar(char ch,int n);
{
for(int j=0;j<n;j++)
cout<<ch;
cout<<endl;
}
```

المخرجات:

```
enter a character:+
enter number of times to repeat it:20
++++++++++++++++++++
```

الشرح:

هذا البرنامج عن تمرير المتغيرات كقيم فعليه عوضا عن الثوابت,حيث اولا تم التصريح بتابع جديد اسمه repchar له معاملان اثنان الاول من النوع char ,الثانى من النوع int, ثم قمنا ضمن جسم البرنامج بتعريف متغيرين جديدين سوف ندخلهم اثناء عمل البرنامج اى من قبل المستخدم وهما chin و nin ,كالتالى:

```
cout<<"enter a character:";
cin>>chin;
cout<<"enter number of times to repeat it:";
cin>>nin;
```

ثم يتم تمريرهم الى معاملات التابع ليكونوا بمثابه القيم الفعليه.
ولاحظ ايضا ان هذا التابع تم استدعاءه مره واحده من خلال الامر التالى:

```
repchar(chin,nin);
```

==========

برنامج (رقم.9):

```
#include <iostream.h>
////////////////////////////
void divide(int a,int b=2)
{
int r;
r=a/b;
cout<<r<<endl;
}
////////////////////////////
void main ( )
{
divide(6);
divide(20,4);
}
```

المخرجات:

```
3
5
```

الشرح:

هذا البرنامج عن القيمه الافتراضيه للبارامترات التى يتم تمريرها لمعاملات التابع,حيث فى البدايه قمنا بدمج التعريف والتصريح للتابع الذى يدعى divide ,وهذا التابع له عدد 2 معامل من نفس النوع int ,وجعلنا هناك قيمه افتراضيه للمتغير الثانى تساوى 2,وهذه القيمه الافتراضيه يتم استخدامها فى حاله عدم اكتمال القيم الفعليه التى ستمرر الى معاملات التابع ضمن صيغه الاستدعاء.
اى عند امر الاستدعاء الاول لهذا التابع divide(6) سوف نستخدم تلك القيمه الافتراضيه.
وعند امر الاستدعاء الثانى للتابع divide(20,4) لن نستخدم تلك القيمه الافتراضيه.

==========

برنامج (رقم.10):

```
#include <iostream.h>
```

```
////////////////////////////////////
void duplicate(int& a,int& b,int& c)
{
a*=2;
b*=2;
c*=2;
}
////////////////////////////////////
void main ( )
{
int x=1;
int y=3;
int z=7;
duplicate(x,y,z);
cout<<"x="<<x<<endl;
cout<<"y="<<y<<endl;
cout<<"z="<<z<<endl;
}
```

المخرجات:

```
x=2
y=6
z=14
```

الشرح:

هذه البرنامج عن التمرير بواسطه المرجع.
حيث فى البدايه قمنا بدمج التعريف والتصريح بالتابع الذى يدعى duplicate والذى يحتوى على ثلاث معاملات من النوع int, واسماء المعاملات الثلاث على الترتيب هم a ثم b ثم c كالتالى:

```
void duplicate(int& a,int& b,int& c)
```

ولكن الشئ الوحيد والجديد والغريب هو وجود علامه الجمع المنطقى & فى السطر الاول التى تدل على التمرير بواسطه المرجع.
ثم قمنا ضمن جسم البرنامج بتعريف ثلاث متغيرات جدد من النوع int وتهيئتهم بقيم ما كالتالى:

```
int x=1;
int y=3;
int z=7;
```

وبعد ذلك قمنا باستدعاء التابع duplicate مع ثلاث قيم فعليه سبق وقد عرفناها كالتالى:

```
duplicate(x,y,z);
```

==========

برنامج (رقم.11):

```
include <iostream.h
////////////////////////////////////////////
void intfrac(float n , float& intp , float& fracp)
{
long temp=static_cast<long>(n);
intp=static_cast<float>(temp);
fracp=n-intp;
}
////////////////////////////////////////////
void main ( )
{
float number;
float intpart;
float fracpart;
cout<<"enter the number:";
cin>>number;
intfrac(number,intpart,fracpart);
```

```
cout<<"the integer part:"<<intpart<<endl;
cout<<"the fractional part:"<<fracpart<<endl;
}
```

المخرجات:

```
enter the number:3.4
the integer part:3
the fractional part:0.4
```

الشرح:

هذا البرنامج عندما تدخل له رقم عشرى يفصله اليك الى جزءين الجزء الحقيقى والجزء العشرى,وهو مثال ايضا على التمرير بواسطه المرجع.

حيث فى البدايه قمنا بدمج التعريف والتصريح بالتابع الذى يدعى intfrac والذى يحتوى على ثلاث معاملات من النوع float, واسماء المعاملات الثلاث على الترتيب هم n ثم intp ثم fracp كالتالى:

```
void intfrac(float n , float& intp , float& fracp)
```

ولكن الشئ الوحيد والجديد والغريب هو وجود علامه الجمع المنطقى & فى السطر الاول التى تدل على التمرير بواسطه المرجع.

ثم قمنا ضمن جسم البرنامج بتعريف ثلاث متغيرات جدد من النوع float كالتالى:

```
float number;
float intpart;
float fracpart;
```

وبعد ذلك قمنا باستدعاء التابع intfrac كالتالى:

```
intfrac(number,intpart,fracpart);
```

ولاحظ ان المتغير number يتم الحصول عليه من قبل المستخدم كالتالى:

```
cout<<"enter the number:";
cin>>number;
```

اما المتغيران intpart و fracpart فيتم الحصول عليهم بواسطه التابع نفسه.

============

برنامج (رقم.12):

```
#include <iostream.h>
//////////////////////////////////////////////
void order(int& numb1,int& numb2)
{
if(numb1>numb2)
{
int temp;
temp=numb1;
numb1=numb2;
numb2=temp;
}
}
//////////////////////////////////////////////
void main ( )
{
int n1;
int n2;
int n3;
int n4;
n1=99;
n2=11;
n3=22;
n4=88;
order(n1,n2);
order(n3,n4);
cout<<"n1="<<n1<<endl;
cout<<"n2="<<n2<<endl;
```

```
cout<<"n3="<<n3<<endl;
cout<<"n4="<<n4<<endl;
}
```

المخرجات:

```
n1=11
n2=99
n3=22
n4=88
```

الشرح:

هذا البرنامج مجرد تطبيق على التمرير بواسطه المرجع,حيث قمنا بدمج التعريف والتصريح بالتابع الجديد وهو order ,هذا التابع يحتوى على عدد 2 معامل من النوع int, وهما numb1 و numb2, هذا التابع تمرر له قيمتين اثنين فاذا كانت القيمه الاولى اكبر من القيمه الثانيه فاننا نقوم بالتبديل بينهما,وكالعاده الشئ الجديد الذى يميز التمرير بواسطه المعاملات هو وجود علامه الجمع المنطقى & ضمن السطر الخاص بالتعريف والتصريح بالتابع كالتالى:

```
void order(int& numb1,int& numb2)
```

ثم قمنا ضمن جسم البرنامج بتعريف وتهيئه اربع متغيرات جدد بقيم معينه كالتالى:

```
int n1;
int n2;
int n3;
int n4;
n1=99;
n2=11;
n3=22;
n4=88;
```

اما عن الجزء الخاص باستدعاء التابع كالتالى:

الحاله الاولى:نلاحظ ان 99 اكبر من 11 وعليه فسوف نبدل القيمتين.

```
order(n1,n2);
```

الحاله الثانيه:نلاحظ ان 22 اقل من 88 وعليه فلن نبدل القيمتين.

```
order(n3,n4);
```

============

برنامج (رقم.13):

```
#include <iostream.h>
////////////////////////////////////////
void prevnext(int x , int& prev , int& next)
{
prev = x-1;
next = x+1;
}
////////////////////////////////////////
void main ( )
{
int x;
int y;
int z;
cout<<"enter the value of x:";
cin>>x;
prevnext(x,y,z);
cout<<"previous="<<y<<endl;
cout<<"next="<<z<<endl;
}
```

المخرجات:

```
enter the value of x:100
previous=99
next=101
```

الشرح:

هذا البرنامج مجرد تطبيق على التمرير بواسطه المرجع الذى يميزه وجود علامه الجمع المنطقى & , وفكره البرنامج انه عندما يتم ادخال عدد ما من قبل المستخدم ولتكن(100)فانه يتم تمريره الى التابع prevnext الذى يقوم بدوره وهى طباعه العدد الذى يسبق(99) والعدد الذى يلى (101)الرقم المدخل, ولان الرقم السابق واللاحق يتم ايجادها داخل التابع ونحتاج لهما خارج التابع فاننا استخدما التمرير بواسطه المرجع.
حيث قمنا بدمج التعريف والتصريح بالتابع prevnext كالتالى:

```
void prevnext(int x , int& prev , int& next)
```

اما ضمن جسم البرنامج فقد عرفنا ثلاث متغيرات جدد من النوع int وهم x و y و z, وهم ايضا ما سيتم تمريره الى معاملات التابع الثلاث x وprev و next كالتالى:

```
int x;
int y;
int z;
```

============

برنامج (رقم.14):

```
#include <iostream.h>
/////////////////////////////////////////////////////////////////////
struct distance
{
int feet;
float inches;
};
/////////////////////////////////////////////////////////////////////
void engldisp(distance dd)
{
cout<<dd.feet<<"'-"<<dd.inches<<"\""<<endl;
}
////////////////////////////////////////////////////////////////////
void main ( )
{
distance d1;
distance d2;
cout<<"enter feet:";
cin>>d1.feet;
cout<<"enter inches:";
cin>>d1.inches;
cout<<"enter feet:";
cin>>d2.feet;
cout<<"enter inches:";
cin>>d2.inches;
cout<<"d1=";
engldisp(d1);
cout<<"d2=";
engldisp(d2);
return;
}
```

المخرجات:

```
enter feet:6
enter inches:4
enter feet:5
enter inches:4.25
d1=6'-4"
d2=5'-4.25"
```

الشرح:

هذا البرنامج مجرد تطبيق على التمرير بواسطه التراكيب.

حيث فى البدايه قمنا بعمل تركيب جديد يدعى distance, هذا التركيب يحتوى على عدد 2 عنصر,الاول هو feet نوعه int, الثانى هو inches نوعه float, وهذا هو الجزء الاول من البرنامج كالتالى:

```
struct distance
{
int feet;
float inches;
};
```

اما التابع الموجود بالبرنامج فهو يدعى engldisp يحتوى على معامل وحيد فقط,ووظيفته هى اننا بمجرد ما ان نمرر له قيم معينه بالفيت والانش فانه سوف يطبعهم فى صوره معينه تم تحديدها.

```
void engldisp(distance dd)
{
cout<<dd.feet<<"'-"<<dd.inches<<"''"<<endl;
}
```

ولكن فى هذا البرنامج سوف نمرر له التركيب الذى انشأناه فى البدايه كمعاملات ليقوم التابع السابق بالعمل عليها,وهذا هو الجزء الثانى من البرنامج.

اما الجزء الثالث من البرنامج فيحتوى على تعريف لمتغيرين جديدين من النوع او التركيب distance هما d1 و d2 كالتالى:

```
distance d1;
distance d2;
```

ثم تم استدعاء التابع مرتين اثنين كالتالى:

```
engldisp(d1);
engldisp(d2);
```

برنامج (رقم.15):

```
#include <iostream.h>
/////////////////////////////////////////////////////////////////////
void ibstokg(float pounds)
{
float kilograms;
kilograms=0.453592*pounds;
cout<<kilograms;
}
////////////////////////////////////////////////////////////////////
void main ( )
{
float ibs;
cout<<"enter weight in pounds:";
cin>>ibs;
cout<<"weight in kilograms:";
ibstokg(ibs);
return;
}
```

المخرجات:

```
enter weight in pounds:182
weight in kilograms:82.5537
```

الشرح:

البرنامج باختصار شديد يطلب منى الوزن بالباوند
ويطبع لى الوزن بالكيلو جرام وذلك من خلال استخدام التابع ibstokg, اما عن استدعاء التابع فقد تم كالتالى:

```
ibstokg(ibs);
```

============

- الاصناف والاهداف فى لغه C++ :-
- تحديد الصنف:

يتخذ تعريف الصنف الصيغه التاليه:

```
class class_name
{
```

```
//body of class
private:
//data and private functions
public:
//data and public functions
};
```

حيث:

class_name يرمز الى اسم الصنف وهو من اختيار المستخدم.
Body of class يرمز الى جسم الصنف وهو يتألف من جزئين وهما:
القسم الخاص private الذى يحتوى على عناصر المعطيات.
القسم العام public الذى يحتوى على توابع العضويه.

بالنسبه للقسم الخاص:

يبتدأ بالكلمه private متبوعه بنقطتين رأسيتين ويحتوى هذا القسم عاده على عناصر معطيات الصنف وقد يحتوى على توابع عضويه خاصه ولكن هذا قليل الاستخدام.

بالنسبه للقسم العام:

يبتدأ بالكلمه public متبوعه بنقطتين رأسيتين ويحتوى هذا القسم عاده على توابع عضويه الصنف وقد يحتوى على عناصر معطيات خاصه ولكن هذا قليل الاستخدام.

============

لاحظ ما يلى:

1- ان المعطيات الخاصه يمكن الوصول اليها من ضمن الصنف,بينما التوابع العامه يمكن الوصول اليها من خارج الصنف.
2- جسم كل تابع عضويه يكون محصورا ضمن قوسين كبيرين.
3- تقوم توابع العضويه فى الصنف بعمليات عامه تتلخص فى الاعداد والاستنتاج للمعطيات المخزنه فى الصنف.
4- ان استخدام الصنف يتم بكل بساطه حيث بمجرد تحديد الصنف يمكن ضمن جسم البرنامج تعريف اهداف من هذا الصنف.

============

المركبات:

1. **ان المركب عباره عن تابع عضويه يتم استدعاؤه تلقائيا بمجرد تشكيل او استدعاء هدف.**
2. **ان اسم المركب يجب ان يكون نفس اسم الصنف.**
3. **ان المركب ليس له قيم مرجعيه وليس له معاملات.**

============

المدمرات:

1. **ان المدمر عباره عن تابع عضويه يتم استدعاؤه تلقائيا بمجرد تدمير هدف.**
2. **ان اسم المدمر يجب ان يكون نفس اسم الصنف.**
3. **ان المدمر ليس له قيم مرجعيه وليس له معاملات.**

============

امثله محلوله:

برنامج (رقم.1):

```
#include <iostream.h>
////////////////////////////////////////////////////
class smallobj
{
private:
int somedata;
public:
void setdata(int d)
{
somedata=d;
}
void showdata( )
{
cout<<"data is:"<<somedata<<endl;
}
};
```

```
/////////////////////////////////////////////////////
void main ( )
{
smallobj s1;
smallobj s2;
s1.setdata(1066);
s1.showdata( );
s2.setdata(1776);
s2.showdata( );
return;
}
```

المخرجات:

```
data is:1066
data is:1776
```

الشرح:

هذا البرنامج مجرد مثال على موضوع الاصناف والاهداف,حيث فى البدايه قمنا بتعريف صنف اسمه smallobj , هذا الصنف له عنصر معطيات وحيد من النوع int, ويسمى somedata كالتالى:

```
int somedata;
```

هذا الصنف ايضا يحتوى على عدد 2 تابع عضويه من النوع void, وهما setdata و showdata .

وظيفه تابع العضويه الاول:تهيئه قيمه معينه.

```
void setdata(int d)
{somedata=d;}
```

وظيفه تابع العضويه الثانى: اظهار تلك القيمه التى سبق ان هيأتها.

```
void showdata( )
{cout<<"data is:"<<somedata<<endl;}
```

ثم قمنا ضمن جسم البرنامج بالتصريح عن هدفين جدد وهما s1,s2 وهما من الصنف السابق smallobj كالتالى:

```
smallobj s1;
smallobj s2;
```

اما عن استدعاء توابع العضويه فقد تم كالتالى:

```
s1.setdata(1066);
s1.showdata( );
s2.setdata(1776);
s2.showdata( );
```

===========

برنامج (رقم.2):

```
#include <iostream.h>
/////////////////////////////////////////////////////
class part
{
private:
int  modelnumber;
int  partnumber;
float cost;
public:
void setpart(int mn,int pn,float c)
{
modelnumber=mn;
partnumber=pn;
cost=c;
}
void showpart ( )
{
cout<<"model:"<<modelnumber<<endl;
```

```
cout<<"part:"<<partnumber<<endl;
cout<<"cost:"<<cost<<endl;
}
};
//////////////////////////////////////////////////
void main ( )
{
part p1;
p1.setpart(6244,373,217.55);
p1.showpart( );
return;
}
```

المخرجات:

```
model:6244
part:373
cost:217.55
```

الشرح:

هذا البرنامج مجرد مثال على موضوع الاصناف والاهداف,حيث فى البدايه قمنا بتعريف صنف اسمه part, هذا الصنف له عدد 3 عناصر معطيات اثنان منهم من النوع int بينما الثالث من النوع float كالتالى:

```
int   modelnumber;
int   partnumber;
float cost;
```

هذا الصنف يحتوى ايضا على عدد 2 تابع عضويه من النوع void .

وظيفه التابع الاول:تهيئه قيم معينه.

```
void setpart(int mn,int pn,float c)
{modelnumber=mn;
partnumber=pn;
cost=c;}
```

وظيفه التابع الثانى:اظهار تلك القيم والتى سبق ان هيأتها.

```
void showpart ( )
{cout<<"model:"<<modelnumber<<endl;
cout<<"part:"<<partnumber<<endl;
cout<<"cost:"<<cost<<endl;}
```

ثم قمنا ضمن جسم البرنامج بالتصريح عن هدف واحد من هذا الصنف وهو يدعى p1 وفق السطر التالى:

```
part p1;
```

اما عن استدعاء توابع العضويه فقد تم كالتالى:

```
p1.setpart(6244,373,217.55);
p1.showpart( );
```

===========

برنامج (رقم.3):

```
#include <iostream.h>
//////////////////////////////////////////////////
class distance
{
private:
int   feet;
float inches;
public:
void setdist(int ft,float in)
{
feet=ft;
inches=in;
}
void getdist( )
```

```
{
cout<<"enter feet:";
cin>>feet;
cout<<"enter inches:";
cin>>inches;
}
void showdist( )
{
cout<<feet<<"'-<<inches<<"'''"<<endl;"
}
};
/////////////////////////////////////////////////////
void main ( )
{
distance d1;
distance d2;
d1.setdist(11,6.25);
d2.getdist( );
cout<<"the 1st distance:";
d1.showdist( );
cout<<"the 2nd distance:";
d2.showdist( );
return;
}
```

المخرجات:

```
enter feet:10
enter inches:4.75
the 1st distance:11'-6.75"
the 2nd distance:10'-4.75"
```

الشرح:

هذا البرنامج مجرد مثال على موضوع الاصناف والاهداف,حيث فى البدايه قمنا بتعريف صنف اسمه distance ,هذا الصنف له عدد 2 عناصر معطيات الاول من النوع int بينما الثانى من النوع float كالتالى:

```
int   feet;
float inches;
```

هذا الصنف يحتوى ايضا على عدد 3 تابع عضويه من النوع void .

وظيفه التابع الاول:هى تهيئه عدد 2قيمه معينه لتزويد عنصرى المعطيات الاثنين.

```
void setdist(int ft,float in)
{feet=ft;
inches=in;}
```

وظيفه التابع الثانى: هى الحصول على 2 قيمه من قبل المستخدم لتزويد عنصرى المعطيات الاثنين.

```
void getdist( )
{cout<<"enter feet:";
cin>>feet;
cout<<"enter inches:";
cin>>inches;}
```

وظيفه التابع الثالث: هى اظهار القيمتين المخزونه فى عنصرى المعطيات الاثنين.

```
void showdist ( )
{cout<<feet<<"'-<<inches<<"'''"<<endl;}"
```

ثم قمنا ضمن جسم البرنامج بالتصريح عن هدفين اثنين من هذا الصنف وهما d1,d2 وذلك وفق ما يلى:

```
distance d1;
distance d2;
```

اما عن استدعاء توابع العضويه فقد تم كالتالى:

```
d1.setdist(11,6.25);
```

```
d2.getdist( );
d1.showdist( );
d2.showdist( );
```

===========

برنامج (رقم.4):

```
#include <iostream.h>
/////////////////////////////////////////////////////
class counter
{
private:
unsigned int count;
public:
counter( ): count(0)
{cout<<"I am a Constructor!!"<<"also,I am prepared by
eng: a.sala7!!";
cout<<endl;}
void inc_count( )
{++count;}
void get_count( )
{cout<<count<<endl;}
};
/////////////////////////////////////////////////////
void main( )
{
counter c1;
counter c2;
cout<<"c1:";
c1.get_count( );
cout<<"c2:";
c2.get_count( );
c1.inc_count( );
c2.inc_count( );
cout<<"c1:";
c1.get_count( );
cout<<"c2:";
c2.get_count( );
return;
}
```

المخرجات:

```
I am a Constructor!!also,I am prepared by eng: a.sala7!!
I am a Constructor!!also,I am prepared by eng: a.sala7!!
c1=0
c2=0
c1=1
c2=1
```

الشرح:

هذا المثال على موضوع المركبات,ويمكن ان تقول ايضا ان المركب عباره عن تابع عضويه يتم استدعاؤه تلقائيا بمجرد تشكيل او استدعاء هدف,ولتعلم ان اسم المركب يجب ان يكون نفس اسم الصنف,ولتعلم ايضا ان المركب ليس له قيم مرجعيه وليس له معاملات.

فقد قمنا بتعريف صنف اسمه counter, وهذا الصنف له عنصر معطيات وحيد من النوع unsigned int وهو count كالتالى:

unsigned int count;

وهذا الصنف ايضا له عدد 3 تابع عضويه حيث:

التابع الاول: counter هو مركب اصلا والذى سيقوم بتهيئه اهدافه دائما بالقيمه صفر بمجرد تشكيل هذه الاهداف.

التابع الثانى: inc_count وظيفته هى اضافه مقدار 1 الى العداد.
التابع الثالث: get_count وظيفته هى طباعه القيمه الحاليه للعداد.
ثم قمنا ضمن جسم البرنامج بالتصريح عن هدفين وهما c1,c2 من الصنف السابق counter كالتالى:

```
counter c1;
counter c2;
```

ولاحظ ان بمجرد تشكيل هذه الاهداف فان العداد سيقوم باعطاءهم قيمه صفر, ومن ثم قمنا باستدعاء التوابع باستخدام الاوامر التاليه:

```
c1.get_count( );
c2.get_count( );
c1.inc_count( );
c2.inc_count( );
```

===========

برنامج (رقم.5):

```
#include <iostream.h>
/////////////////////////////////////////////////////
class foo
{
private:
int data;
public:
foo( ): data(0)
{cout<<"I am a constructor!!"<<"also,I am prepared by eng:
a.sala7!!";
cout<<endl;}
~foo( )
{cout<<"I am a destructor!!"<<"also,I am prepared by eng: a.sala7!!";
cout<<endl;}
};
```

الشرح:

هذا المثال على موضوع المدمرات,ويمكن ان تقول ان المدمر عباره عن تابع عضويه ينفذ تلقائيا بمجرد تدمير هدف,ولتعلم ان اسم المدمر يجب ان يكون نفس اسم الصنف,ولتعلم ايضا ان المدمر ليس له قيمه مرجعيه وليس له معاملات.
فقد قمنا بتعريف صنف يدعى foo .
هذا الصنف له عنصر معطيات وحيد من النوع int وهو data كالتالى:

```
int data;
```

هذا الصنف له عدد 2 تابعى عضويه الاول هو مركب والثانى هو مدمر.
صيغه المركب كالتالى:

```
foo( ): data(0)
```

صيغه المدمر كالتالى:

```
~foo( )
```

ولاحظ وجود العلامه المميزه للدلاله على المدمر وهى ~ .

===========

برنامج (رقم.6):

```
#include <iostream.h>
/////////////////////////////////////////////////////
class distance
{
private:
int   feet;
float inches;
public:
distance( ):
feet(0),inches(0.0)
```

```
{cout<<"I am a Constructor!!"<<"also,I am
prepared by eng: a.sala7!!";
cout<<endl;}
distance(int ft,float in):feet(ft),inches(in)
{cout<<"I am a Constructor!!"<<"also,I am
prepared by eng: a.sala7!!";
cout<<endl;}
void getdist( )
{cout<<"enter feet:";
cin>>feet;
cout<<"enter inches:";
cin>>inches;}
void showdist( )
{cout<<feet<<"'"<<inches<<"'"'<<endl;}
void adddist(distance,distance);
};
/////////////////////////////////////////////////////
void main( )
{
distance d1;
distance d3;
distance d2(11,6.25);
d1.getdist( );
d3.adddist(d1,d2);
cout<<"the 1st distance:";
d1.showdist( );
cout<<"the 2nd distance:";
d2.showdist( );
cout<<"the 3rd distance:";
d3.showdist( );
return;
}
/////////////////////////////////////////////////////
void distance::adddist(distance d2,distance d3)
{
inches=d2.inches+d3.inches;
feet=0;
if(inches>=12.0)
{
inches-=12.0;
++feet;
}
feet+=d2.feet+d3.feet;
}
```

المخرجات:

```
I am a Constructor!!"<<"also,I am prepared by eng: a.sala7!!
I am a Constructor!!"<<"also,I am prepared by eng: a.sala7!!
I am a Constructor!!"<<"also,I am prepared by eng: a.sala7!!
enter feet:17
enter inches:5.75
d1=17'-5.75"
d2=11'-6.25"
d3=29'-0"
```

الشرح:

هذا البرنامج يستعرض لنا مزايا جديده من الاصناف مثل التحميل الزائد للمركبات,تعريف توابع عضويه خارج الصنف,تعريف اهداف كمعاملات للتوابع.
بدايه فقد قمنا بتعريف صنف يدعى distance .
هذا الصنف له عدد 2 عنصر معطيات الاول من النوع int, والثانى من النوع float كالتالى:

```
int   feet;
float inches;
```

هذا الصنف له عدد 5 تابع عضويه ,حيث:
تابع العضويه الاول: عباره عن مركب ولكنه بدون اى معاملان.

```
distance( ):feet(0),inches(0.0)
```

تابع العضويه الثانى: عباره عن مركب ولكنه به معاملان اثنان.

```
distance(int ft,float in):feet(ft),inches(in)
```

تابع العضويه الثالث: getdist وظيفته الحصول على عدد 2 قيمه من قبل المستخدم وذلك لتزويد عنصرى المعطيات الاثنين.
تابع العضويه الرابع: showdist وظيفته اظهار القيمتين المخزونه فى عنصرى المعطيات الاثنين.
تابع العضويه الخامس: adddist فقد تم التصريح به داخل الصنف اما التعريف به فقد تم خارج الصنف.
والان قمنا ضمن جسم البرنامج بالتصريح عن هدفين جديدين وهما d1,d3 من الصنف الجديد distance كالتالى:

```
distance d1;
distance d3;
```

وايضا صرحنا عن هدف ثالث وهو d2 من نفس الصنف ولكننا اعطيناه قيمتين اوليين بالفيت والانش هما 11 و 6.25 على الترتيب كالتالى:

```
distance d2(11,6.25);
```

وباستخدام التابع الثالث getdist فاننا سوف نحصل على قيم d1 بالفيت والانش من قبل المستخدم كالتالى:

```
void getdist( )
{cout<<"enter feet:";
cin>>feet;
cout<<"enter inches:";
cin>>inches;}
```

اما عن القيم الخاصه ب d2 فاننا نحصل عليها من التابع الخامس adddist باستخدام الامر التالى:

```
d3.adddist(d1,d2);
```

اما عن التابع الرابع showdist فعن طريقه تظهر كل القيم السابقه على الشاشه كالتالى:

```
d1.showdist( );
d2.showdist( );
d3.showdist( );
```

============

المصفوفات فى لغه C++ :-

الصف:-

هو تتابع من المتغيرات كلها من نفس النوع,وهذه المتغيرات تسمى عناصر الصف,يتم ترقيم هذه العناصر بالتتابع 0 ثم 1 ثم 2 .....الخ,هذه الارقام تسمى الفهرس او الدلائل,وهى التى تحدد مكان العنصر فى الصف..

============

ملاحظات:-

1- الصف هو مصفوفه احاديه البعد.
2- عناصر المصفوفه تكون من نفس النوع.

============

مقارنه بين التراكيب والمصفوفات:-

عناصر التراكيب يمكن ان تكون من انواع مختلفه , بينما عناصر المصفوفه تكون كلها من نفس النوع , وهذا هو جوهر الاختلاف بين التراكيب والمصفوفات.

============

الصيغه العامه لتعريف المصفوفه:-

لتعريف مصفوفه ما نستخدم التعبير التالى:

```
Array_type array_name [array_size];
```

حيث:
Array_type تمثل نوع عناصر المصفوفه.
Array_name تمثل اسم المصفوفه.
Array_size تمثل عدد عناصر المصفوفه وبالطبع يجب ان يكون عدد صحيح ثابت.

مثال:

```
Int array[4];
Float array[10];
```

============

ادخال عناصر المصفوفه:-

يقصد بادخال عناصر المصفوفه اعطاء كل عنصر من عناصر المصفوفه قيمته المطلوبه والمطلوبه لنوعه,وسوف نستخدم الصيغه التاليه:

```
Array_name [number of one] = value of one;
```

حيث:

Array_name تمثل اسم المصفوفه.

Number of one تمثل رقم او ترتيب عنصر المصفوفه سواء كان 0 او 1 او 2 .....الخ.

Value of one تمثل القيمه المراد اسنادها للعنصر المحدد ترتيبه او رقمه.

مثال:

```
Array[4]=14;
Array[9]=12;
```

============

ادخال قيمه عناصر مصفوفه باستخدام الحلقات التكراريه:-

بدلا من تعريف كل عنصر على حدى يمكن استخدام احد الحلقات التكراريه كالتالى:

```
For(int i=0;i<3;i++)
Cout<<"enter the number:";
Cin>>array[i];
```

وهذا يعنى:

ان هناك عدد 3 عناصر وانت ستدخل قيمهم بنفسك على الترتيب,حيث اخذت عناصر المصفوفه الترتيب 0 ثم 1 ثم 2 وهذه بمثابه الدلائل او الفهرس.

============

تعريف حجم المصفوفه:-

يفضل احيانا تعريف حجم المصفوفه كثابت كالتالى:

```
Const int SIZE=10;
```

وذلك بفرض ان عدد عناصرها يساوى 10 ,وهذه الطريقه تتيح لنا التعديل فى حجم المصفوفه من خلال هذا السطر فقط دون الاضطرار الى التعديل فى كافه انحاء البرنامج,وتذكر ان كتابه اسم الثابت تتم عاده باحرف كبيره وذلك للدلاله على ان هذا الاسم يمثل ثابت ولا يمكن التغيير فى قيمته ابدا.

============

ملاحظات هامه:-

1- من اجل تعريف وتهيئه مصفوفه لا حاجه لتحديد حجمها,وذلك لان المترجم سيقوم بنفسه بمعرفه عدد عناصرها.

2- اذا عرفنا مصفوفه ذات حجم معين وهيأناها بقيم عددها اقل من الحجم المحدد للمصفوفه,فعندئذ فان العناصر التى لم نحدد قيمتها سوف تعتبر صفرا.

============

المصفوفات متعدده الابعاد:-

لتعريف مصفوفه ثنائيه الابعاد نستخدم الصيغه التاليه:

```
Array_type array_name[row_count]
[col_count];
```

حيث:

Array_type تمثل نوع عناصر المصفوفه.

Array_name تمثل اسم المصفوفه.

Row_count تمثل عدد اسطر او صفوف المصفوفه.

Col_count تمثل عدد اعمده المصفوفه.

مثال:

```
int array [1] [2];
```

============

الوصول الى عناصر مصفوفه ما:-

من اجل الوصول الى عنصر ما بين اى مصفوفه ثنائيه الابعاد فان ذلك يتم عن طريق اسم المصفوفه والدلائل كالتالى:

```
Array [d] [m]
```

حيث:
Array تمثل اسم المصفوفه.
d , m تمثل دلائل المصفوفه.
ومن اجل الوصول الى عنصر ما بين اى مصفوفه ثلاثيه الابعاد فان ذلك يتم عن طريق اسم المصفوفه والدلائل كالتالى:

```
Array [i] [j] [k]
```

حيث:
Array تمثل اسم المصفوفه.
i , j , k تمثل دلائل المصفوفه.
============

اهم المؤثرات ووظائفها:-

اولا:- setiosflags(ios::fixed)
هذا المؤثر يستخدم من اجل منع ظهور الارقام الناتجه بشكل اسى.

ثانيا:- setiosflags(ios::showpoint)
هذا المؤثر يستخدم من اجل ظهور الارقام الناتجه من اى عمليه فى برنامج ما مع خانات عشريه حتى لو كان هذا الرقم الناتج ليس عشرى.

ثالثا:- setpricision(n)
هذا المؤثر يستخدم من اجل تخصيص مسافه مقدارها n بعد الفاصله العشريه.

وقد تجد ايا من هذه المؤثرات الرائعه فى اى برنامج ما!!

فكان علينا ان نذكر بعضا منها ووظائفها!!
============

امثله محلوله:

برنامج (رقم.1):

```
#include <iostream.h>
//////////////////////////////////////////////////
void main( )
{
const int SIZE=4;
int array[4];
for (int i=0;i<4;i++)
{
cout<<"enter the number:";
cin>>array[i];
}
cout<<"the groups of our array:"<<endl;
for (int j=0;j<4;j++)
{
cout<<"array["<<j<<"]="<<array[j]<<endl;
}
return;
}
```

المخرجات:

```
enter the number:1
enter the number:2
enter the number:3
enter the number:4
:the groups of our array
array[0]=1
array[1]=2
array[2]=3
array[3]=4
```

الشرح:

هذا البرنامج مثال بسيط على المصفوفات,حيث اولا قمنا ضمن جسم البرنامج بتعريف حجم المصفوفه كثابت من خلال السطر التالى:

```
const int SIZE=4;
```

وثانيا تم تعريف مصفوفه تدعى array ,وعناصرها عددهم 4 عناصر,وجميعهم من النوع int من خلال السطر التالى:

```
int array[4];
```

ثم استخدمنا حلقه تكراريه لادخال قيم عناصر المصفوفه الاربع من قبل المستخدم كالتالى:

```
for (int i=0;i<4;i++)
{
cout<<"enter the number:";
cin>>array[i];
}
```

ثم بعد ذلك استخدمنا حلقه تكراريه اخرى طباعه عناصر المصفوفه بالترتيب كالتالى:

```
for (int j=0;j<4;j++)
{
cout<<"array["<<j<<"]="<<array[j]<<endl;
}
```

===============

برنامج (رقم2.):

```
include <iostream.h
//////////////////////////////////////////////////
void main( )
{
const int SIZE=6;
double sales[SIZE];
cout<<"please enter the widget sales for 6 days:"<<endl;
for (int i=0;i<SIZE;i++)
{
cin>>sales[i];
}
double total=0;
for (int j=0;j<6;j++)
{
total+=sales[j];
}
double average=total/SIZE;
cout<<"the average value:"<<average<<endl;
return;
}
```

المخرجات:

```
please enter the widget sales for 6 days:
100
150
650.45
123.45
225.50
168.50
the average value:239.86
```

الشرح:

يمكن القول ان هذا البرنامج مثال حى على ادخال مستخدم ما لقيم مبيعات سلعه تجاريه لمده سته ايام ثم الحصول على المتوسط الحسابى للمبيعات.

ولنفرض ان لدينا مصفوفه عدد عناصرها 6 عناصر وعند ادخالهم فان البرنامج يقوم بحساب المتوسط الحسابى لهم.

حيث اولا قمنا ضمن جسم البرنامج بتعريف حجم المصفوفه كثابت من خلال السطر التالى:

```
const int SIZE=6;
```

وثانيا تم تعريف مصفوفه تدعى sales وعناصرها عددهم 6 عناصر,وجميعهم من النوع float من خلال السطر التالى:

```
double sales[SIZE];
```

ثم استخدمنا حلقه تكراريه لادخال عناصر المصفوفه الست من قبل المستخدم كالتالى:

```
for (int i=0;i<SIZE;i++)
{
cin>>sales[i];
}
```

ثم عرفنا متغير جديد من النوع double ويدعى total واعطيناه قيمه ابتدائيه تساوى صفر وفق السطر التالى:

```
double total=0;
```

ثم عرفنا متغير جيدي اخر يدعى average ونوعه double وهو يساوى ناتج قسمه total على SIZE وفق السطر التالى:

```
double average=total/SIZE;
```

واستخدمنا حلقه تكراريه لايجاد المجموع الكلى للعناصر كالتالى:

```
for (int j=0;j<6;j++)
{
total+=sales[j];
}
```

===============

برنامج (رقم.3):

```
#include <iostream.h>
/////////////////////////////////////////////////////
void main( )
{
int month;
int day;
int total_days;
}; int days_per_month[12]={ 31,28,31,30,31,30,31,31,30,31,30,31
cout<<"enter month(1 to 12):";
cin>>month;
cout<<"enter day(1 to 31):";
cin>>day;
total_days=day;
for(int j=0;j<month-1;j++)
{
total_days+=days_per_month[j];
}
cout<<"total days from start of year to given date:";
cout<<total_days<<endl;
return;
}
```

المخرجات:

```
enter month(1 to 12):3
enter day(1 to 31):11
total days from start of year to given date:70
```

الشرح:

هذا البرنامج يقوم بحساب عدد الايام التى مرت من بدايه السنه حتى تاريخ معين انت تدخله بنفسك اى من قبل المستخدم.

حيث قمنا ضمن جسم البرنامج بتعريف ثلاث متغيرات يمثلوا الشهر واليوم واجمالى الايام وجميعهم من النوع int كالتالى:

```
int month;
int day;
int total_days;
```

ثم عرفنا مصفوفه تسمى days_per_month, وعدد عناصرها 12 عنصر, وجميعهم من النوع int, وقيمه كل عنصر تتحدد بعدد الايام الموجود فى كل شهر من شهور السنه
كشهر فبراير 28 يوم وهكذا كالتالى:

```
}; int days_per_month[12]={ 31,28,31,30,31,30,31,31,30,31,30,31
```

ثم الجزء الخاص بادخال الشهر واليوم من قبل المستخدم كالتالى:

```
cout<<"enter month(1 to 12):";
cin>>month;
cout<<"enter day(1 to 31):";
cin>>day;
```

ثم استخدمنا حلقه تكراريه لحساب اجمالى الايام المطلوبه كالتالى:

```
for(int j=0;j<month-1;j++)
{
total_days+=days_per_month[j];
}
```

===============

برنامج (رقم.4):

```
#include <iostream.h>
///////////////////////////////////////////
struct part
{
int modelnumber;
int partnumber;
float cost;
};
///////////////////////////////////////////
void main( )
{
const int SIZE=4;
int n;
part part1[SIZE];
for(n=0;n<4;n++)
{
cout<<endl;
cout<<"enter modelnumber:";
cin>>part1[n].modelnumber;
cout<<"enter partnumber:";
cin>>part1[n].partnumber;
cout<<"enter cost:";
cin>>part1[n].cost;
}
cout<<endl;
for(n=0;n<4;n++)
{
cout<<"model:"<<part1[n].modelnumber<<endl;
cout<<"part:"<<part1[n].partnumber<<endl;
cout<<"cost in $:"<<part1[n].cost<<endl;
}
return;
}
```

المخرجات:

```
enter modelnumber:1
enter partnumber:2
enter cost:3
```

```
enter modelnumber:4
enter partnumber:5
enter cost:6

enter modelnumber:7
enter partnumber:8
enter cost:9

enter modelnumber:10
enter partnumber:11
enter cost:12

model:1
part:2
cost in $:3

model:4
part:5
cost in $:6
```

الشرح:

هذا البرنامج مثال حى على مصفوفه التراكيب,فهذا البرنامج يحتوى على تركيب لنوع جديد تم تصميمه وهو part,وهذا التركيب يحتوى على عدد 3 عناصر الاول يمثل الرقم وهو من النوع int,الثانى يمثل الكميه وهو من النوع int,الثالث يمثل السعر وهو من النوع float,كالتالى:

```
struct part
{
int modelnumber;
int partnumber;
float cost;
};
```

ثم عرفنا متغير جديد يدعى n من النوع int وهذا المتغير سوف نستخده فى الحلقات التكراريه التى سننشأها, وذلك وفق السطر التالى:

```
int n;
```

وتم تعريف حجم المصفوفه كثابت وفق السطر التالى:

```
const int SIZE=4;
```

وتم تعريف مصوفه اسمها part1, وعناصر تلك المصفوفه عددهم 4 عناصر, وجميعهم من نوع التركيب part, وذك وفق السطر التالى:

```
part part1[SIZE];
```

ثم تم استعمال حلقه تكراريه لادخال عناصر المصفوفه من قبل المستخدم كالتالى:

```
for(n=0;n<4;n++)
{
cout<<endl;
cout<<"enter modelnumber:";
cin>>part1[n].modelnumber;
cout<<"enter partnumber:";
cin>>part1[n].partnumber;
cout<<"enter cost:";
cin>>part1[n].cost;
}
```

وتم ايضا بعد ذلك استعمال حلقه تكراريه لطباعه عناصر المصفوفه بالترتيب كالتالى:

```
for(n=0;n<4;n++)
{
cout<<"model:"<<part1[n].modelnumber<<endl;
cout<<"part:"<<part1[n].partnumber<<endl;
cout<<"cost in $:"<<part1[n].cost<<endl;
```

```
}
```

================

برنامج (رقم5):

```
#include <iostream.h>
//////////////////////////////////////////
class stack
{
private:
enum {MAX=10};
int st[MAX];
int top;
public:
stack( )
{top=0;}
void push(int var)
{st[++top]=var;}
int pop( )
{return st[top--];}
};
//////////////////////////////////////////
void main( )
{
stack s1;
s1.push(11);
s1.push(22);
cout<<"1:"<<s1.pop( )<<endl;
cout<<"2:"<<s1.pop( )<<endl;
s1.push(33);
s1.push(44);
s1.push(55);
s1.push(66);
cout<<"3:"<<s1.pop( )<<endl;
cout<<"4:"<<s1.pop( )<<endl;
cout<<"5:"<<s1.pop( )<<endl;
cout<<"6:"<<s1.pop( )<<endl;
return
}
```

المخرجات:

```
1:22
2:11
3:66
4:55
5:44
6:33
```

الشرح:

هذا البرنامج كمثال على استخدام المصفوفات كبيانات معطاه داخل الصنف.
حيث بدأنا بتعريف صنف جديد يدعى stack, هذا الصنف يحتوى على ثلاث عناصر معطيات,هذا الصنف ايضا يحتوى على ثلاث عناصر توابع عضويه.
بالنسبه لعنصر المعطيات الاول:فيه تعريف لحجم مصفوفه كثابت يساوى عشره كالتالى:

```
enum {MAX=10};
```

بالنسبه لعنصر المعطيات الثانى:فتم فيه التعريف بمصفوفه جديده اسمها st عدد عناصرها 10 عناصر,وجميعهم من النوع int كالتالى:

```
int st[MAX];
```

بالنسبه لعنصر المعطيات الثالث:فتم فيه التعريف بمتغير جديد من النوع int ويدعى top كالتالى:

```
int top;
```
بالنسبه لتابع العضويه الاول:فهو مركب يحمل نفس اسم الصنف الا وهو stack وتم ضمن هذا التابع عمل قيمه باتدائيه للمتغير top وهى تساوى صفرا كالتالى:

```
stack( )
{top=0;}
```
بالنسبه لتابع العضويه الثانى:فندخل به قيمه داخل الصنف.

بالنسبه لتابع العضويه الثالث:لاخراج القيمه السابقه عن الصنف اى ارجاعها.

وقمنا ضمن جسم البرنامج بتعريف متغير جديد من نوع الصنف ويدعى s1 كالتالى:

```
stack s1;
```
وقمنا بعد ذلك باستدعاء تابعى العضويه الثانى والثالث حسب الحاجه.

===============

برنامج (رقم.6):

```
#include <iostream.h>
//////////////////////////////////////////////////////
class distance
{
private:
int   feet;
float inches;
public:
void getdist( )
{
cout<<"enter feet:";
cin>>feet;
cout<<"enter inches:";
cin>>inches;
}
void showdist( )
{
cout<<feet<<"'-"<<inches<<"'''"<<endl;
}
};
//////////////////////////////////////////////////////
void main( )
{
distance d[100];
int n=0;
char ans;
do{
cout<<"enter number of distance:"<<n+1;
d[n++].getdist( );
cout<<"enter another (y/n)?";
cin>>ans;
}while(ans!='n')
for(int j=0;j<n;j++)
{
cout<<"distance number"<< j+1<<" is:";
d[j] .showdist( );
}
cout<<endl;
return;
};
```

المخرجات:

```
enter number of distance:1
```

```
enter feet:5
enter inches:4
enter another (y/n)?y
enter number of distance:2
enter feet:6
enter inches:2.5
enter another (y/n)?y
enter number of distance:3
enter feet:5
enter inches:10.75
enter another (y/n)?n
distance number1 is:5'-4"
distance number2 is:6'-2.5"
distance number3 is:5'-10.75"
```

الشرح:

هذا البرنامج مثال على مصفوفه الاهداف, حيث عرفنا فى البدايه صنف جديد يدعى distance .
هذا الصنف يحتوى على عنصرى معطيات وهما الفييت والانش,الاول من النوع int, الثانى من النوع float .
هذا الصنف ايضا يحتوى على عنصرى تابعى عضويه حيث:
الاول: وظيفته الحصول على 2 قيمه معينه لتزويد عنصرى المعطيات الاثنين.

```
void getdist( )
{
cout<<"enter feet:";
cin>>feet;
cout<<"enter inches:";
cin>>inches;
}
```

والثانى: وظيفته اظهار القيمتين المخزونه فى عنصرى المعطيات الاثنين.

```
void showdist( )
{
cout<<feet<<"'-"<<inches<<"\""<<endl;
}
```

ثم قمنا ضمن جسم البرنامج بتعريف مصفوفه تسمى d, عدد عناصرها 100 عنصر,وجميعهم من نوع الصنف distance كالتالى:

```
distance d[100];
```

ثم تعريف وتهيئه متغيرين جديدن كالتالى:

```
int n=0;
char ans;
```

===============

تم بحمد الله,,